OLIVER KOLOSSOSKI

PROPOSTA DE ALGORITMO BASEADO EM DEFLAÇÃO POLINOMIAL PARA DETERMINAÇÃO DE RAÍZES DE POLINÔMIOS

CURITIBA

DEZEMBRO, 2012

OLIVER KOLOSSOSKI

PROPOSTA DE ALGORITMO BASEADO EM DEFLAÇÃO POLINOMIAL PARA DETERMINAÇÃO DE RAÍZES DE POLINÔMIOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Matemática aplicada, pelo Departamento de Matemática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Luiz Carlos Matioli

CURITIBA

DEZEMBRO, 2012

Ministério da Educação
Universidade Federal do Paraná
Setor de Ciências Exatas/Departamento de Matemática
Programa de Pós-Graduação em Matemática Aplicada - PPGMA

ATA DA 45ª DEFESA DE DISSERTAÇÃO DE MESTRADO

Ao vinte e dois dias do mês de fevereiro de 2013, no Anfiteatro B, Prédio PC do Setor de Ciências Exatas, da Universidade Federal do Paraná, foi instalada pelo Professor Luiz Carlos Matioli, a Banca Examinadora para a quadragésima quinta Dissertação de Mestrado em Matemática Aplicada. Estiveram presentes ao Ato, professores, alunos e visitantes.

A banca examinadora, homologada pelo Colegiado do Programa de Pós-Graduação em Matemática Aplicada, ficou constituída pelos professores: Prof. Dr. Alagacone Sri Ranga, da Universidade Estadual de São Paulo; Prof. Dr. Marcelo Muniz Silva Alves, do Departamento de Matemática-UFPR, e o Prof. Dr. Luiz Carlos Matioli, Orientador da dissertação, a quem coube a presidência dos trabalhos.

Às quatorze horas, a banca iniciou seus trabalhos, convidando o candidato **Oliver Kolossoski** a fazer a apresentação do tema da dissertação intitulada "Proposta de Algoritmo Baseado em Deflação Polinomial para Determinação de Raízes de Polinômios". Encerrada a apresentação, iniciou-se a fase de argüição pelos membros participantes. Após a argüição, a banca com pelo menos 03 (três) membros, reuniu-se para apreciação do desempenho do pós-graduando,

A banca considerou que o pós-graduando fez uma apresentação com a necessária concisão. A Dissertação apresenta contribuição à área de estudos e não foram registrados problemas fundamentais de estrutura e redação, resultando em plena e satisfatória compreensão dos objetivos pretendidos.

Tendo em vista a dissertação e a argüição, os membros presentes da banca decidiram pela sua aprovação.

Curitiba, 22 de fevereiro de 2013.

Prof. Dr. Luiz Carlos Matioli
Presidente

Prof. Dr. Alagacone Sri Ranga
Titular

Prof. Dr. Marcelo Muniz Silva Alves
Membro

Ministério da Educação
Universidade Federal do Paraná
Setor de Ciências Exatas/Departamento de Matemática
Programa de Pós-Graduação em Matemática Aplicada - PPGMA

PARECER DA BANCA EXAMINADORA

Após a apresentação, a banca deliberou pela aprovação da dissertação do candidato **Oliver Kolossoski** devendo para tanto incorporar as sugestões feitas pelos membros da banca, no prazo estabelecido pelo regimento correspondente.

Curitiba, 22 de fevereiro de 2013.

Prof. Dr. Luiz Carlos Matioli
Presidente

Prof. Dr. Alagacone Sri Ranga
Titular

Prof. Dr. Marcelo Muniz Silva Alves
Membro

Agradecimentos

Agradeço à Deus, por estar a meu lado. Agradeço sobretudo ao meu orientador prof. Dr. Luiz Carlos Matioli, pois sem ele esta dissertação, bem como [1] não teriam sido realizados. Agradeço aos meus pais e aos meus amigos pelo constante apoio. Agradeço também à CAPES/REUNI, pelo apoio financeiro. Gostaria de dedicar este trabalho aos meus amigos Claudio, Fábio Mandira, e Bruno Lago.

Sumário

Lista de Símbolos

n, m, k, j, i, n_0 . *Números naturais, que podem significar índices de laços, coeficientes ou apenas números dependendo do contexto.*

x, α, z, β ..... *Variáveis reais ou complexas, dependendo do contexto.*

$M, N, L, \epsilon, \delta$ ..... *Constantes positivas.*

f, p, ϕ, Φ ..... *Diferentes funções, tendo suas regras explicitadas no primeiro uso.*

V, X, Ω ..... *Conjuntos cuja propriedade é explicitada no primeiro uso.*

$\mathbb{N}, \mathbb{R}, \mathbb{C}$ ..... *Respectivamente os conjuntos dos números naturais, reais e complexos.*

$\mapsto$ ..... *Símbolo designado para se explicitar regras de funções.*

$\overset{\mathbb{N}'}{\to}$ *Convergência no conjunto* $\mathbb{N}'$*. Quando a convergência é em* $\mathbb{N}$*, não se põe o índice em cima da seta.*

$\{x_k\}_{k\in\mathbb{N}}$ ..... *Sequência dos números* x_k *com* k *variando em* $\mathbb{N}$*.*

$B(x, \delta)$ *Conjunto dos pontos* $y \in \mathbb{R}^n$ *(ou* y *em outro conjunto dependendo do contexto) tais que* $\|x - y\| < \delta$*, para alguma norma* $\|.\|$*.*

$\max(X), \min(X), \sup(X), \inf(X)$. *Respectivamente valores máximo, mínimo, supremo e ínfimo do conjunto* X*.*

$\liminf(X)$ ..... *O menor dos pontos de aderência do conjunto* X*.*

$\mathcal{P}_n(z)$ ..... *Conjunto dos polinômios de grau* n *na variável* z*.*

$\binom{m}{k}$ ..... *Combinação de* m *elementos,* k *à* k*.*

$\sum_{j=0}^{n} a_j$, ou $\sum_{j=0}^{n} a_j$ ..... *Soma dos valores* a_j*, com* j *variando em* $\mathbb{N} \cup \{0\}$ *de* 0 *à* n*.*

$(a_0, \ldots a_n)^t$ ..... *o vetor em* $\mathbb{R}^{n+1}$ *(ou* $\mathbb{C}^{n+1}$*, dependendo do contexto) com entradas* $a_0, \ldots a_n$*.*

$A(1:j, n)$ ..... *A submatriz da matriz* A*, cujas entradas são* $a_{1n}, \ldots a_{jn}$*.*

$floor(j)$ ..... *O maior número inteiro que não é estritamente maior que* j*.*

$f(X)$, onde f é uma função com domínio contendo o conjunto X ..... *Conjunto dos pontos* $y \in imf$*, tais que existe* $x \in X$ *satisfazendo* $f(x) = y$*.*

$O(n)$ *Dizemos que uma função* $f(x)$ *pertence à* $O(n)$ *se existem* $x_0 \in Domf$ *e* $M > 0$ *tais que para todo* $x > x_0$ *vale* $f(x) \leq Mn$*. Por tradição, ao invés de denotar* $f \in O(n)$*, denotamos* $f = O(n)$*.*

Resumo

Calcular zeros de polinômios é um problema de vasta aplicabilidade na ciência, porém, é um problema difícil de se resolver para polinômios de grau alto, o que faz dos métodos numéricos um ótimo modo de se atacá-lo. Um outro problema já antigo e mesmo assim, bastante relacionado que aparece em matemática aplicada é o de determinar os autovalores de uma dada matriz. Contudo, dependendo do método numérico utilizado para se calcular as raízes de um polinômio, um outro problema que é usual aparecer é como se deflacionar o polinômio, este é um passo que deve ser incluso no algoritmo que encontra as raízes, e pode causar efeitos na estabilidade do mesmo, e também em seu desempenho. Neste trabalho apresentamos um método novo para deflacionar m vezes um polinômio, dada uma raiz de multiplicidade m do mesmo, e após isso, apresentamos um novo método para calcular raízes de polinômios baseando-se na ideia do algoritmo de deflação apresentado.

Palavras-chave: Deflação polinomial; Raízes de polinômios; Multiplicidade.

Abstract

Calculating zeros of polynomials is a problem which appear in a vast number of applications in science. However, it is a very hard problem to solve specially for polynomials of high degree, which makes the numerical methods a great way to approach it. A very old problem, also related to that, which appears in Applied Mathematics is finding the eigenvalues of a given matrix. However, dependending on the numerical method used for calculating the roots of a polynomial, another problem that ocurrs is how to deflate the obtained resulting polynomial, such deflation must be incorporated in the final step of the root-finding algorithm, and can cause stability issues on it, also compromising its performance. In this work we present a new method to deflate m times a polynomial, given a root of multiplicity m of it, and after that, we give a new method for calculating roots of polynomials basing on the idea of the deflation method presented before.

Keywords: Polynomial deflation; Roots of polynomials; Multiplicity.

1 INTRODUÇÃO

O trabalho trata de um importante problema da Análise Numérica, que é o de calcular raízes de um polinômio de coeficientes reais. No entanto, como a maioria dos métodos acabam encontrando apenas uma ou duas raízes de cada vez, um outro problema que está vinculado é o de deflacionar o polinômio inicial para se obter um novo polinômio e desta forma aplicar o método novamente a fim de se obter as raízes do polinômio inicial. Para que isto fique mais claro, considere um algoritmo que calcula uma raiz a cada vez, e que o aplicamos ao polinômio $p(z) = (z-2)^2(z-1)$. Se o algoritmo nos fornece a raiz $\alpha = 2$, então devemos dividir $p(z)$ por $(z-2)$ para obter $\overline{p}(z) = (z-2)(z-1)$, e então aplicar novamente o método ao polinômio $\overline{p}(z)$ para encontrar uma nova raiz. No entanto se soubermos de antemão que a multiplicidade da raiz $\alpha = 2$ é 2, então ao invés de dividir $p(z)$ duas vezes sucessivas por $(z-2)$, podemos dividir o polinômio por $(z-2)^2$.

Descreveremos no capítulo 3 um método que generaliza esta ideia para raízes de multiplicidade m qualquer. No capítulo 4, usaremos a ideia deste método de deflação para desenvolver dois métodos para calcular raízes de polinômios, veremos que por ter usado a ideia do método de deflação, não precisaremos efetuar deflação para aplicar o método desenvolvido. Os dois métodos visarão calcular respectivamente uma raiz simples e uma raiz dupla, desde que o polinômio possua uma. Veremos

mais adiante como verificar se o polinômio em questão tem raízes com multiplicidade ou não. Por fim, no capítulo 5 faremos considerações sobre os dois métodos elaborados. Mas antes de tudo, no capítulo 2 estabeleceremos alguns resultados que nos serão úteis ao longo do trabalho exposto.

2 ALGUNS RESULTADOS IMPORTANTES

2.1 Alguns Resultados Importantes

Nesta seção apresentaremos alguns resultados que serão necessários para o desenvolvimento do nosso trabalho, não necessariamente estando relacionados uns com os outros. Assumimos que o leitor está familiarizado com a notação e a teoria de sequências de números reais, se preciso, o leitor pode consultar o apêndice 6 para tal fim. Começaremos demonstrando a conhecida relação de Stifel.

TEOREMA 2.1.1 (RELAÇÃO DE STIFEL) *Para quaisquer* $k < n \in \mathbb{N} - \{0\}$*, vale*

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$$

Demonstração. Com efeito,

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \frac{(n-1)!}{(k-1)!(n-k)!} + \frac{(n-1)!}{k!(n-1-k)!} =$$

$$= \frac{k(n-1)!}{k!(n-k)!} + \frac{(n-k)(n-1)!}{k!(n-k)!} = \frac{(k+n-k)(n-1)!}{k!(n-k!)} = \frac{n(n-1)!}{k!(n-k)!} = \binom{n}{k}.$$

■

A expansão de Taylor de uma função $f(z)$ ao redor de um ponto z_0 de seu domínio vale para qualquer função infinitamente diferenciável, e pode ter sua dedução

consultada em qualquer livro de cálculo ou análise, veja por exemplo [10]. A expressão é dada por

$$f(z) = \sum_{j=0}^{\infty} \frac{f^{(j)}(z_0)}{j!}(z - z_0)^j$$

Onde $f^{(j)}$ representa a j-ésima derivada de f. No caso de f ser uma função polinomial, temos que a soma é finita, pois pondo por exemplo $p(z) = a_0 z^n + \cdots + a_{n-1}z + a_n$, vemos claramente que se $j > n$, então $p^{(j)} \equiv 0$, e assim a série se reduz à:

$$p(z) = \sum_{j=0}^{n} \frac{p^{(j)}(z_0)}{j!}(z - z_0)^j.$$

Considere dois polinômios $P(z)$ e $b(z)$, o primeiro sendo de grau n e o segundo de grau m. Considere ainda que eles são dados pelas seguintes expressões: $p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1}z + a_n$, $B(z) = b_0 z^m + b_1 z^{m-1} + \cdots + b_{m-1}z + b_m$. Então se multiplicarmos os dois polinômios, obteremos um outro polinômio $pB(z)$ de grau $n+m$, que pode ser escrito da forma $pB(z) = c_0 z^{n+m} + c_1 z^{n+m-1} + \cdots + c_{n+m-1}z + c_{n+m}$, e seus coeficientes c_k podem ser escritos em função de a_j e b_i. Com efeito, igualando-se as expressões, obtemos:

$$(a_0 z^n + \cdots + a_{n-1}z + a_n)(b_0 z^m + \cdots + b_{m-1}z + b_m) = c_0 z^{n+m} + \cdots + c_{n+m-1}z + c_{n+m}$$

Se aplicarmos a propriedade distributiva devidamente, veremos que cada coeficiente c_k, com $k = 0, \ldots, n+m$ satisfaz $c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_k b_0$, donde temos a seguinte expressão para os coeficientes do produto de dois polinômios:

$$c_k = \sum_{j=0}^{k} a_j b_{k-j}. \tag{2.1}$$

esta relação será usada posteriormente.

Como estaremos lidando com algoritmos, precisaremos da noção de aritmética de ponto flutuante. Apresentaremos aqui alguns destes conceitos. Aritmética de ponto flutuante é uma aritmética digital usada para simular a aritmética dos números reais. Nenhum computador consegue representar os números da reta real, pois estes são infinitos, e a memória de um computador é sempre finita, podendo trabalhar apenas um número finito de elementos.

Sendo assim para cada número real x em $\mathbb{R}$, representaremo-lo por $\bar{x}$, donde $\bar{x}$ é igual a x até um número fixado de casas. Chamaremos este número fixado de t, e nos referiremos a ele como "precisão".

Para que o número $\bar{x}$ seja uma boa aproximação de x, dada nossa precisão, não podemos escrever x de qualquer maneira, como por exemplo, se $x = 100000$, e $t = 5$, escrevendo $\bar{x}$ como as primeiras cinco casas de x, teremos $\bar{x} = 10000$, que é dez vezes menor que x. Com uma precisão de 5, gostaríamos que o erro cometido ao se aproximar x por $\bar{x}$ não excedesse 10^{-4}. Para tal devemos escrever tanto x e $\bar{x}$ em notação científica, isto é, $x = 0.a_1a_2 \ldots a_t \cdots \times 10^e$, e então $\bar{x} = 0.a_1a_2 \ldots a_t \times 10^e$, assim obteremos um erro $|x - \bar{x}| \leq 5 \times 10^{-t}$.

O número e que usamos na representação de x e $\bar{x}$ é chamado de "expoente", e diz a ordem de grandeza do número x "antes" da vírgula em sua representação usual. Este número deve ser limitado, pois não podem haver infinitos números na aritmética de ponto flutuante, sendo assim impomos um limite inferior e superior para e, digamos $L \leq e \leq U$.

Para fins de generalidade, podemos representar $\bar{x}$ em qualquer outra base que não seja 10, ou seja, $\bar{x} = 0.a_1a_2 \ldots a_t \times \beta^e$, com $L \leq e \leq U$, onde β é a base escolhida. Chamamos o conjunto dos possíveis números a serem usados para representar x por

$\bar{x}$ de sistema, e seus elementos de "pontos flutuantes". A aritmética de ponto flutuante é então a aritmética dos números reais restrita à este sistema.

Apresentaremos uma definição mais formal de sistema de pontos flutuantes:

DEFINIÇÃO 2.1.1 *Um sistema de pontos flutuantes é um subconjunto* $F \subset \mathbb{R}$*, tal que dados 4 números* $\beta, t \in \mathbb{N}$*, e* $L, U \in \mathbb{Z}$*, então* F *pode ser descrito como* $\{x \in \mathbb{R} : x = 0.a_1a_2 \ldots a_t \times \beta^e\}$*, com* $a_i \in \{0, 1, \ldots, \beta - 1\}$*,* $a_1 \neq 0$*, e* $L \leq e \leq U$*.*

Note que o sistema F pode ser caracterizado apenas pelos quatro números, ou seja (β, t, L, U). Um valor típico correspondente ao sistema de pontos flutuantes para um computador seria $F = (10, 32, -32, 32)$.

Fixado um sistema F, os erros cometidos ao se aproximar x por $\bar{x}$, conforme comentado anteriormente, não excedem $u = 1/2 \times \beta^{1-t}$. Este número u é chamado de "unidade de arredondamento" e é usado para se medir os erros cometidos em um determinado algoritmo. Tal análise geralmente é feita junto com a análise de custo operacional do algoritmo, isto é, calcular o número de operações, em geral, apenas somas e/ou multiplicações, para se executar o algoritmo. Ambas as análises estão intimamente relacionadas.

Os erros cometidos para se fazer as operações, naturalmente se acumulam, podendo um algoritmo estar sujeito a problemas de erros de arrendondamento conhecidos da análise numérica, como "underflow" e "overflow". O primeiro deles é o erro cometido por se aproximar um número pequeno por zero. Note que em $(10, 32, -32, 32)$ qualquer número menor que 10^{-33} é considerado zero. Desta forma, para se efetuar a operação $10^{30} \times 10^{-34}$, cujo resultado é 10^{-4}, dependendo da forma conforme esta é realizada, o computador fará $10^{30} \times 0$, resultando em 0. Este é o problema conhecido

como “underflow”.

Já o “overflow”, mais perigoso que o “underflow” é o erro no computador quando tentamos representar em (β, t, L, U) um número maior que $0.\beta - 1\beta - 1 \ldots \beta - 1 \times \beta^U$. Quando isso acontece, em geral o computador ao invés de aproximar tal número pelo maior possível, o que seria o análogo do que acontece com o “underflow”, ele retorna uma mensagem de erro e pára o algoritmo. Tal erro também pode ser cometido por acumulação de erros de arredondamento, como por exemplo quando efetuamos divisões por números muito pequenos.

Estes dois problemas justificam a importância da análise de erros de arrendondamento, bem como análise de custos operacionais para algoritmos. O leitor interessado em um estudo mais aprofundado pode consultar, por exemplo [12].

Precisaremos posteriormente, também, do conceito de algoritmo. Não chegaremos a definir algoritmos formalmente, apenas pensaremos que um algoritmo é um método numérico que para cada “ponto inicial” x^0 gera uma sequência $\{x^k\}$. Pode ser pensado em uma função $A : \mathbb{R}^n \mapsto S$, onde S é o conjunto de todas as sequências. Existem definições mais sofisticadas de algoritmos, veja por exemplo [11].

Seja $\mathcal{P}$ uma propriedade qualquer. Concentraremos então em buscar um algoritmo que para cada ponto inicial gere uma sequência que convirja para um ponto que satisfaz $\mathcal{P}$. Faremos então a seguinte definição:

DEFINIÇÃO 2.1.2 *Dados uma propriedade $\mathcal{P}$, um conjunto $\Omega \subset \mathbb{R}^n$, e $x \in \mathbb{R}^n$, dizemos que x é um ponto desejável com relação à $\mathcal{P}$ se satisfaz $\mathcal{P}$, e dizemos que é viável com relação à Ω se $x \in \Omega$.*

Se estiver claro no contexto, apenas falaremos em ponto viável e ponto desejável. Sendo assim, dado um problema matemático, digamos "*encontre um ponto de* Ω *que satisfaz* $\mathcal{P}$", procuraremos encontrar algoritmos que o resolvam no seguinte sentido: Para cada ponto inicial x^0 encontrar uma sequência que converge a um ponto de Ω desejável com relação à propriedade $\mathcal{P}$.

A noção de algoritmo ficará mais clara com um exemplo deste, que daremos a seguir. Uma notação que será usada é a da avaliação de Horner de um dado polinômio. Considere um polinômio com coeficientes reais da forma

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n.$$

Então fixado $\overline{z} \in \mathbb{C}$, para se avaliar $p(\overline{z})$, devemos efetuar $j+1$ multiplicações para cada termo $a_{n-j} z^j$, o que pode resultar em perda de informação, ou até mesmo "underflow" e/ou "overflow". Para se evitar isso existe o algoritmo de Horner para a avaliação de um polinômio. Tal algoritmo é o seguinte

ALGORITMO 2.1.1 *Dado* $p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$.

$p(z) = a_0$;

Para $i = 1 \ldots n$

$$p(z) = z p(z) + a_i;$$

Fim

No fim do algoritmo, teremos a avaliação polinomial que queremos, gastando apenas n somas e multiplicações.

Deixamos aqui fixada uma notação, que será útil posteriormente, denotaremos a avaliação de um polinômio $p(z) = \sum_{j=0}^{n} a_j z^{n-j}$ em um escalar α por $H(a_{0,\ldots,n}, \alpha)$.

Para nosso próximo resultado, precisaremos da noção de função Lipschitziana.

DEFINIÇÃO 2.1.3 *Uma função $f : \mathbb{R} \mapsto \mathbb{R}$ é dita Lipschitziana se existe uma constante $C > 0$ tal que para todos $x, y \in \mathbb{R}$ tem-se:*

$$|f(x) - f(y)| \leq C|x - y|,$$

sendo C a constante de Lipschitz.

PROPOSIÇÃO 2.1.2 *Seja $f : \mathbb{R} \mapsto \mathbb{R}$ uma função Lipschitziana, e $\{\alpha_k\}$ uma sequência convergente à α, tal que $\alpha_{k+1} = f(\alpha_k)$. Então α é um ponto fixo de f, isto é, $f(\alpha) = \alpha$.*

Demonstração. De fato, seja C a constante de Lipschitz. Como α_k é convergente, é de Cauchy, donde para todo $\epsilon > 0$, existe $n_\epsilon \in \mathbb{N}$ tal que $|\alpha_{k+1} - \alpha_k| < \frac{\epsilon}{C}$, quando $k > n_\epsilon$.

Sendo assim, tem -se que para $k > n_\epsilon$, $|f(\alpha_{k+1}) - f(\alpha_k)| \leq C|\alpha_{k+1} - \alpha_k| < \epsilon$.

Desta forma, $|f(\alpha_{k+1}) - f(\alpha_k)| \to 0$ quando $n \to \infty$, e como $f(\alpha_k) = \alpha_{k+1}$, tem-se então que $|f(\alpha_k) - \alpha_k| \to 0$, logo deve-se ter que $f(\alpha) = \alpha$.

■

O próximo resultado é uma versão do conhecido teorema do ponto fixo de Banach para a reta real.

TEOREMA 2.1.3 (TEOREMA DO PONTO FIXO) *Seja $f : \mathbb{R} \mapsto \mathbb{R}$ contínua tal que para*

todo x, y *em* $\mathbb{R}$ *tem-se* $|f(x) - f(y)| \leq C|x - y|$*, com* $0 < C < 1$*. Então existe um único ponto fixo* $\bar{x} \in \mathbb{R}$ *com* $f(\bar{x}) = \bar{x}$*.*

Demonstração.

Unicidade: Sejam x e y dois pontos fixos de f, então $|x - y| = |f(x) - f(y)| \leq C|x - y|$, como $C < 1$ tal inequação só é possível se $|x - y| = 0$, logo $x = y$.

Existência: Considere a sequência real $x_{k+1} = f(x_k)$. Usando a desigualdade triangular para módulos diversas vezes temos que

$$|x_{k+l} - x_k| \leq \sum_{j=1}^{l-1} |x_{k+j} - x_{k+j-1}|,$$

E pela hipótese, temos que $|x_{k+j} - x_{k+j-1}| \leq C^{k+j-1}|x_1 - x_0|$. Portanto, combinando as duas inequações, temos que

$$|x_{k+l} - x_k| \leq \sum_{j=1}^{l-1} C^{k+j-1}|x_1 - x_0| \leq$$

$$\leq |x_1 - x_0|C^k \sum_{j=1}^{\infty} C^{j-1} = |x_1 - x_0|C^k \frac{1}{1-C} \mapsto 0.$$

Segue que a sequência $x_{k+1} = f(x_k)$ é de Cauchy, e como é real, converge para algum ponto $\bar{x} \in \mathbb{R}$.

Por fim, tomando o limite na definição da sequência, e usando a continuidade de f, temos que $f(\bar{x}) = \bar{x}$. ∎

A seguir introduziremos conceitos auxiliares que nos permitirão definir a “bacia de atração”, cuja noção será utilizada no capítulo 5.

DEFINIÇÃO 2.1.4 *Dizemos que o conjunto dos números complexos estendidos, isto é* $\mathbb{C} \cup \{\infty\}$ *é a Esfera de Riemann, e denotamo-la por* $\hat{\mathbb{C}}$*.*

Podemos definir algumas operações para o elemento adicional ∞, em analogia com as operações usuais em $\mathbb{C}$, estas são: $z\infty = \infty \; \forall z \in \hat{\mathbb{C}}$ (Incluíndo o próprio ∞), e ainda $\frac{z}{\infty} = 0$, e $\frac{z}{0} = \infty \; \forall z \in \mathbb{C}$, estas últimas operações não estão definidas para $z = \infty$. Observamos que as operações $\infty \pm \infty$ não estão definidas.

Geometricamente podemos pensar na esfera de Riemann como "o plano complexo envolvendo uma esfera", sendo o pólo sul desta estando no ponto $z = 0$, e para cada ponto $z \in \mathbb{C}$, traçamos uma linha de z ao pólo norte da esfera, e o encontro desta linha com a esfera é a representação deste em $\hat{\mathbb{C}}$. Os "pontos no infinito", isto é, $z = \infty$ são levados no pólo norte da esfera.

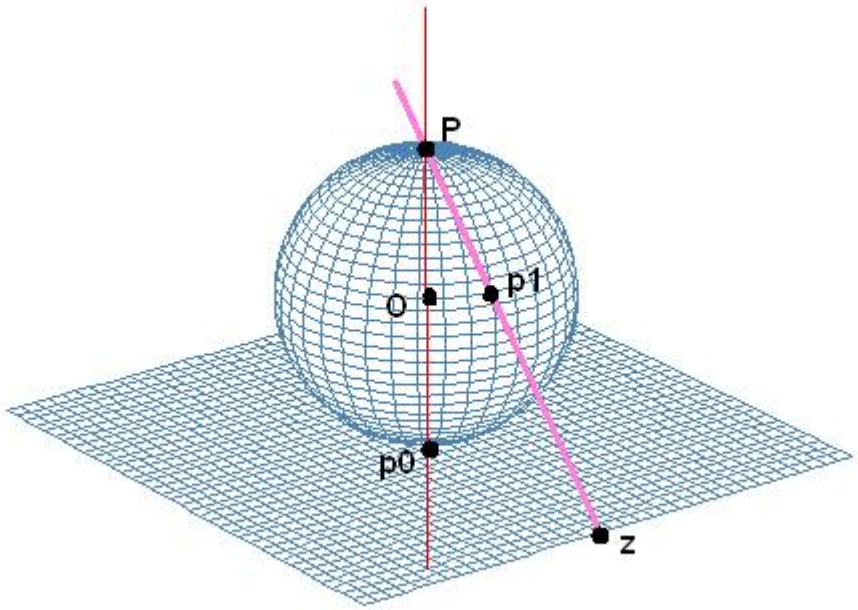


FIGURA 1: A esfera de Riemann e a correspondência da projeção sobre os pontos do plano

Na figura 1 temos a origem O, e o pólo norte P da esfera. Traçamos os segmentos ligando um ponto z do plano complexo à P, e também o segmento ligando a origem O à P, para efeitos de ilustração. Os correspondentes em $\hat{\mathbb{C}}$ $P1$ e $P0$ também

são marcados na figura.

Uma função racional em $\mathbb{C}$, da forma $f(z) = \frac{g(z)}{h(z)}$ pode ser estendida à uma função contínua em $\hat{\mathbb{C}}$, por exemplo, com nossas operações definidas acima, a função $f(z) = \frac{z+1}{z-10}$ é tal que $f(10) = \infty$, e $f(\infty) = 1$. Por abuso de linguagem chamamos uma extensão deste tipo de "Aplicação racional em $\hat{\mathbb{C}}$".

DEFINIÇÃO 2.1.5 *Seja $f : \hat{\mathbb{C}} \mapsto \hat{\mathbb{C}}$ uma aplicação racional na esfera de Riemann $\hat{\mathbb{C}}$, para cada ponto $z \in \hat{\mathbb{C}}$ definimos a órbita de z por $orb(z) = \{z, f(z), \dots f^n(z) \dots\}$.*

DEFINIÇÃO 2.1.6 *Dizemos que z_0 é um ponto fixo de período m de f se m é o menor número inteiro tal que $f^m(z_0) = z_0$.* Isto é o mesmo que dizer que z_0 é um ponto fixo de f^m na noção usual.

Dizemos ainda que um ponto z_0 é atrator se $|f'(z_0)| < 1$, repulsor, se $|f'(z_0)| > 1$ e neutro se $|f'(z_0)| = 1$. Se a derivada de f em z_0 é nula, z_0 é dito super-atrator.

Uma órbita, ou mais geralmente um subconjunto de $\hat{\mathbb{C}}$ é dito atrator se todos seus pontos o são, e periódico de período m (com respeito à f) se todos seus pontos o forem. Estamos prontos agora para definir a bacia de atração:

DEFINIÇÃO 2.1.7 *Se O é uma órbita atratora e periódica de período m com respeito à f, a bacia de atração é o conjunto aberto $A \subset \hat{\mathbb{C}}$ de todos os pontos $z \in \hat{\mathbb{C}}$ tal que as iterações $f^m(z), f^{2m}(z), \dots$ convergem para algum ponto de O.*

Para se ter uma ideia menos abstrata do uso de bacias de atração, daremos um exemplo, se tratando de seu uso para estudo do comportamento de métodos numéricos. Neste caso pense em $f(z)$ como a função iteração de um método iterativo, então estaremos interessados em encontrar pontos de equilíbrio desta função, pois

queremos que a iteração $x^{k+1} = f(x^k)$ convirja para algum ponto. Neste caso tais pontos de equilíbrio formam a nossa órbita O. A bacia de atração seria um conjunto ao redor dos pontos deste conjunto tal que se em uma dada iteração, o ponto do passo x^k está na bacia, então o algoritmo convergirá para um dos pontos de O. Podemos então estudar o comportamento do método destacando na bacia os pontos que convergem para um ponto em específico, e os pontos que convergem para um outro. Quanto mais uniforme a bacia, menos "caótico" mostra ser o comportamento do método, isto em termos não minuciosos. Para melhor compreensão destes conceitos, veja a referência [7]. Nesta referência o autor foi o pioneiro em termos de uso de bacia de atração como meio de comparação para métodos numéricos.

3 UM MÉTODO DE DEFLAÇÃO POLINOMIAL

Neste capítulo apresentaremos um algoritmo que reduz para $n - m$ o grau de um polinômio de grau n desde que conhecida uma raiz de multiplicidade m. A ideia do algoritmo é bastante simples, usa apenas o conceito de divisão de polinômios.

Consideraremos aqui polinômios com coeficientes reais da forma

$$p(z) = a_0 z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n = \sum_{i=0}^{n} a_i z^{n-i}. \tag{3.1}$$

onde a_i, $i = 0, \ldots, n$ são números reais.

O capítulo é organizado da seguinte forma: Na seção 3.1 apresentamos o algoritmo para o caso real, ou seja, quando a raiz dada é real. Primeiro apresentamos o resultado para multiplicidade 1 e então generalizamos para multiplicidade m. Na seção 3.2 apresentamos o algoritmo para o caso complexo, quando a raiz considerada é complexa. Na seção 5.1 do capítulo 5 apresentaremos alguns experimentos que ilustram a eficácia do algoritmo desenvolvido neste capítulo e mostram que o algoritmo pode ser utilizado.

3.1 O Caso Real

Nesta seção mostraremos que há uma relação de recorrência entre os coeficientes do polinômio quociente, a qual depende dos coeficientes de $p(z)$, desde que a divisão seja exata. Usaremos desta relação para obter o algoritmo.

A relação é dada pelo seguinte teorema:

TEOREMA 3.1.1 *Considere o polinômio (3.1) dividido por* $d(z) = z^m + c_1 z^{m-1} + \cdots + c_{m-1}z + c_m$ *e o quociente* $q(z) = b_0 z^{n-m} + b_1 z^{n-m-1} + \cdots + b_{n-m-1}z + b_{n-m}$ *com* $n \geq m \geq 1$. *Se a divisão é exata, então* $b_i = a_i - c_1 b_{i-1} - \cdots - c_m b_{i-m}, \quad i = 0, \ldots, n-m$ *com* $b_{-1} = b_{-2} = \cdots = b_{-m} = 0$.

Demonstração. Com efeito, como a divisão é exata, temos que vale $p(z) = d(z)q(z)$. usando a relação 2.1 para os coeficientes do polinômio produto entre $d(z)$ e $q(z)$, temos que:

$$a_i = \sum_{l=0}^{i} c_l b_{i-l}, \quad \forall \quad i = 0, \ldots, n$$

Com $c_0 = 1$. Separando o termo quando $l = 0$, conseguimos:

$$a_i = c_0 b_i + \sum_{l=1}^{i} c_l b_{i-l}$$

Donde usando que $c_0 = 1$ e isolando o coeficiente b_j, temos que:

$$b_i = a_i - \sum_{l=1}^{i} c_l b_{i-l} = a_i - c_1 b_{i-1} - \cdots - c_i b_0.$$

Pondo $b_l = 0$ se $l < 0$, conseguimos a relação desejada para b_i, $\forall\, i = 0, \ldots, n-m$. ∎

Uma vez que este resultado foi provado, podemos usá-lo para obter um algoritmo recursivo para a divisão dos polinômios $p(z)$ pelo fator $(z - \alpha)^m$ que frequente-

mente aparece no problema de deflação de polinômios com raízes com multiplicidade. Isto pode ser feito, pois sabemos que no problema de deflação a divisão é exata. Agora, se α é uma raiz conhecida de multiplicidade m então pelo binômio de Newton, o polinômio divisor tem a seguinte forma

$$(z-\alpha)^m = z^m - \binom{m}{1}\alpha z^{m-1} + \cdots + (-1)^j \binom{m}{j}\alpha^j z^{m-j} + \cdots + (-1)^m \binom{m}{m}\alpha^m$$

que satisfaz a hipótese do teorema 3.1.1 com $c_j = (-1)^j \alpha^j \binom{m}{j}$. Desta forma, podemos deflacionar $p(z)$ se sua raiz encontrada α tem multiplicidade m usando o seguinte algoritmo:

ALGORITMO 3.1.1 *Dados b =* $(0,\ldots,0)^t \in \mathbb{R}^{n+1}$ *e* p *da forma (3.1).*

Para $i = 0,\ldots,n-m$

$b_{m+i} = a_i; \quad c = 0;$

Para $j = 1,\ldots,m$

$c = c + (-1)^{j+1}\alpha^j \binom{m}{j} b_{m+i-j};$

Fim

$b_{m+i} = b_{m+i} + c;$

Fim

O algoritmo acima retorna nas últimas entradas de b os coeficientes do polinômio resultante da divisão de $p(z)$ pelo fator $(z-\alpha)^m$.

3.1.1 Sobre a complexidade do algoritmo

A seguir faremos a análise de custo computacional do algoritmo 3.1.1. O mesmo usa menos operações do que se fizessemos m deflações lineares, isto é, se usássemos o algoritmo tradicional apresentado abaixo para fazer as deflações:

ALGORITMO 3.1.2 *Para* $j = 1, \ldots, m$

Para $i = 1, \ldots, n - j$

$a_i = a_i - \alpha a_{i-1}$;

Fim

Fim

Pode-se ver que para cada valor de j o laço interno custa $n - j$ somas e multiplicações, de modo que o algoritmo 3.1.2 custa $\sum_{j=1}^{m}(n - j) = \frac{m(2n-m-1)}{2}$ somas e multiplicações.

já o algoritmo 3.1.1 usa no laço externo $n - m$ somas, e somando as somas requeridas no laço interno, temos outras m, totalizando n somas. Já para multiplicações, como para se computar os $\binom{m}{j}$ são necessárias $m + j + 1 + (m - j) = 2m + 1$ multiplicações, e para o laço interno $\sum_{j=1}^{m} 2(j + 1)(2m + 1) = m(2m^2 + 7m + 3)$ multiplicações.

Note que o custo computacional requerido pelo algoritmo 3.1.1 depende apenas do valor de m enquanto que o do algoritmo 3.1.2 depende de n. No entanto, o custo do algoritmo proposto só pode ser afirmado menor que o algoritmo 3.1.2 se $m << n$, mais precisamente quando $n > \frac{4m^2+15m+7}{2}$.

Contudo, o algoritmo 3.1.1 ainda pode ter seu número de operações reduzido.

Olhando para o laço em i, vemos que várias operações inócuas estão sendo realizadas, pois não precisamos multiplicar os b_{-i}, uma vez que todos são zero. Um modo fácil de remediar isso é dividir em dois laços: quando $i = 1, ..., m$ fazemos j variar de 1 a i, e quando $i = m + 1, ..., n - m$ fazemos o laço j como mostrado no algoritmo.

Mas o maior custo computacional do algoritmo 3.1.1 é no cálculo de $\binom{m}{j}$. Ora, isto também pode ser melhorado. Primeiro note que só precisamos computar m coeficientes c_i, pois os que dependem de j no laço j são apenas os b_{m+i-j}. Assim, poderíamos fazer um laço até m só para computar os c_i e então multiplicá-los pelos b_{m+i-j}. O bom em fazer isso é que podemos usar a relação de Stifel para encontrar um modo barato de computar c_i usando c_{i-1}. Pela relação de Stifel, temos

$$\binom{m+1}{j+1} = \binom{m}{j+1} + \binom{m}{j}.$$

Então, uma vez conhecido $\binom{m}{j}$, podemos obter $\binom{m}{j+1}$ via

$$\binom{m}{j+1} = \binom{m+1}{j+1} - \binom{m}{j}.$$

Agora, note que

$$\binom{m+1}{j+1} = \frac{(m+1)m!}{(j+1)j!((m+1)-(j+1))!} = \frac{m+1}{j+1}\frac{m!}{(m-j)!j!} = \frac{m+1}{j+1}\binom{m}{j},$$

então pela expressão acima, obtemos $\binom{m}{j+1}$ de

$$\binom{m}{j+1} = (\frac{m+1}{j+1} - 1)\binom{m}{j}.$$

Desta forma, podemos obter c_{i+1} de c_i fazendo

$$c_{i+1} = -\alpha(\frac{m+1}{i+1} - 1)c_i.$$

Agora, levando todas estas modificações em consideração, podemos escrever um novo algoritmo da seguinte forma:

ALGORITMO 3.1.3 *Dados b =* $(0,\ldots,0)^t \in \mathbb{R}^{n+1}$. $\quad c_0 = 0; \quad c_1 = \alpha m$

Para $i = 2,\ldots,m$

$c_i = \alpha(1 - \frac{m+1}{i+1})c_{i-1};$

Fim

Para $i = 1,\ldots,m$

Para $j = 1,\ldots,i$

$c_0 = c_0 + c_j b_{m+i-j}$*;*

Fim

$b_{m+i} = a_i + c_0$*;*

Fim

Para $i = m+1,\ldots,n-m$

$c_0 = 0$*;*

Para $j = 1,\ldots,m$

$c_0 = c_0 + c_j b_{m+i-j}$*;*

Fim

$b_{m+i} = a_i + c_0$*;*

Fim

Agora, contando as operações deste último algoritmo, temos, para o primeiro laço $3(m-1)$ somas e multiplicações, para o segundo, contando as multiplicações, temos $\sum_{i=1}^{m} i + \sum_{i=m+1}^{n-m} m = \frac{m(m+1)}{2} + m(n-2m) = nm + \frac{m}{2} - \frac{3m^2}{2}$ multiplicações, e para somas conseguimos este mesmo número nos laços internos mais $n-m$ nos laços externos, totalizando $nm + \frac{7m}{2} - \frac{3m^2}{2} - 3$ multiplicações e $nm + \frac{5m}{2} - \frac{3m^2}{2} + n - 3$ somas. enquanto se usássemos m deflações, como no algoritmo 3.1.2, teríamos que efetuar $n-1+n-2+\cdots+n-m$ = $\frac{m(2n-m-1)}{2}$ multiplicações, bem como somas.

3.2 O Caso Complexo

Nesta seção mostraremos a versão do algoritmo para o caso complexo, ou seja, para raízes complexas. Os resultados acima não dependem se os coeficientes c_i são reais, o que nos faz acreditar que é possível estabelecer um algoritmo para o caso em que conhecemos uma raiz complexa $\alpha + \beta i$. Neste caso, como estamos considerando que $p(z)$ tem coeficientes reais, podemos dizer que $\alpha - \beta i$ também é uma raiz. Tentaremos, portanto, construir um algoritmo que deflaciona o polinômio dado, levando em conta tanto a raiz encontrada quanto sua conjugada.

É suficiente se encontrarmos uma fórmula para a expressão

$$[(z-(\alpha+\beta i))\,(z-(\alpha-\beta i))]^m \tag{3.2}$$

pois como a divisão é exata, estamos nas condições do teorema 3.1.1. Note que o teorema não demanda que z seja real. Expandindo a expressão (3.2) para $m = 1, 2$, começamos a suspeitar que isto é possível. De fato,

$$[(z-(\alpha+\beta i))\,(z-(\alpha-\beta i))] = \left(z^2 - 2\alpha z + \alpha^2\right) + \beta^2. \tag{3.3}$$

e

$$[(z-(\alpha+\beta i))(z-(\alpha-\beta i))]^2$$

$$=\left(z^4-4\alpha z^3+6\alpha^2 z^2-4\alpha^3 z+\alpha^4\right)+2\beta^2\left(z^2-2\alpha z+\alpha^2\right)+\beta^4. \tag{3.4}$$

No próximo teorema, generalizamos (3.3) e (3.4) para todo m.

TEOREMA 3.2.1 *A seguinte relação é válida:*

$$[(z-(\alpha+\beta i))(z-(\alpha-\beta i))]^m$$

$$=(z-\alpha)^{2m}+\binom{m}{1}\beta^2(z-\alpha)^{2m-2}+\ldots$$

$$+\binom{m}{j}\beta^{2j}(z-\alpha)^{2m-2j}+\cdots+\binom{m}{m-1}\beta^{2m-2}(z-\alpha)^{2m-2(m-1)}+\beta^{2m}.$$

Demonstração. De fato, usando a equação 3.3, temos que

$$[(z-(\alpha+\beta i))(z-(\alpha-\beta i))]^m=\left((z-\alpha)^2+\beta^2\right)^m$$

Expandindo esta última expressão pelo binômio de Newton, conseguimos

$$[(z-(\alpha+\beta i))(z-(\alpha-\beta i))]^m=\sum_{j=0}^{m}\binom{m}{j}(z-\alpha)^{2(m-j)}\beta^{2j}=$$

$$=(z-\alpha)^{2m}+\binom{m}{1}\beta^2(z-\alpha)^{2m-2}+\ldots$$

$$+\binom{m}{j}\beta^{2j}(z-\alpha)^{2m-2j}+\cdots+\binom{m}{m-1}\beta^{2m-2}(z-\alpha)^{2m-2(m-1)}+\beta^{2m}.$$

conforme queríamos demonstrar. ∎

Desta forma podemos escrever o polinômio divisor $d(z)$ da seguinte forma:

$$d(z) = \sum_{j=0}^{m} \sum_{k=0}^{2(m-j)} \beta^{2j} \binom{m}{j} \binom{2(m-j)}{k} ((-1)^k)(\alpha^k) z^{2(m-j)-k} \tag{3.5}$$

Esta soma pode ser computada em dois laços. Contudo note que os coeficientes c_i da fórmula deduzida no teorema 3.1.1 não estão explicitamente dados na equação (3.5). Estes coeficientes podem ser computados separadamente durante os dois laços que usamos para computar $d(z)$. Com isso, podemos obter também um algoritmo para o caso complexo.

O seguinte algoritmo retorna nas últimas entradas de b os coeficientes do polinômio quociente da divisão de $p(z)$ por $[(z-(\alpha+\beta i))\,(z-(\alpha-\beta i))]^m$

ALGORITMO 3.2.1 *Dados* $b = (0, 0, \ldots, 0, a_0, a_1, \ldots, a_{n-m}) \in \mathbb{R}^{n+1}$

Para $k = 1, \ldots, m$

$c_k = 0$;

Para $j = 1, \ldots, 2(m-k)$

$c_k = c_k + \binom{2(m-k)}{j} \alpha^{2(m-k)-j}\,(-1)^j$;

Fim

$c_k = \binom{m}{k} \beta^{2k} c_k$;

Fim

Para $i = 2m+1, \ldots, n+1$

Para $j = 1, \ldots, m$

$b_i = b_i - c_j b_{i-j}$;

Fim

Fim

Aqui os coeficientes c_k são os que fazem o papel dos coeficientes c_k no teorema 3.1.1, mas olhando para a fórmula estabelecida no teorema 3.2.1, usamos dois laços para computá-los, conforme foi apresentado anteriormente. O laço em k computa todos os coeficientes c_k e soma-os, após multiplicados por $\binom{m}{j}\beta^{2k}$, conforme deduzido. Os próximos dois laços, que são independentes, definem b_i pela relação deduzida no teorema 3.1.1, fazendo uso dos c_k computados. Note que como definimos b_i a partir de b_{i-2k-j}, e i começa em $2m+1$, todos os valores de b_{i-2k-j} estarão definidos, e a recursão, por sua vez, também está bem definida.

3.2.1 Novamente Sobre Complexidade

No caso complexo, se efetuarmos a contagem das operações do algoritmo 3.2.1, teremos, para o primeiro laço em j, $12k^2-24mk+12m^2$ multiplicações e $2m-2k$ somas. Somando essas quantias para $k=1,\ldots,m$, teremos $\frac{m^2-m}{2}$ somas, e $4m^3-6m^2+2m$ multiplicações.

Já para os laços restantes, conseguimos, no laço interno em j, m somas e multiplicações, e somando para o laço externo em i, temos $mn-2m^2+m$ somas e multiplicações.

Somando ambas as parcelas, teremos, portanto $mn-\frac{3m^2}{2}+\frac{m}{2}$ somas e $mn+4m^3-8m^2+3m$ multiplicações. Considerando ainda a multiplicação efetuada fora do laço j, e no laço k para computar os c_k, o número de multiplicações é dado por $mn+4^3-8m^2+5m+2k+2$.

Podemos reduzir o número necessário de operações, assim como fizemos no caso real. Podemos economizar operações se evitarmos computar os $\binom{m}{k}$, bem como os $\binom{2(m-k)}{j}$ dos laços onde computamos os c_k. Para fazê-lo, basta usar a relação já desenvolvida

$$c_{k+1} = -\alpha(\tfrac{m+1}{k+1} - 1)c_k.$$

Assim, podemos usar este artifício para computar os c_k isoladamente, bem como para efetuar as multiplicações por $\alpha^{2(m-k)-j}$ e β^{2k} de modo conveniente, como fizemos para o caso real.

Para os dois últimos laços para a computação dos b_i, o único modo de economizar é separar em dois laços, para não efetuar as operações com os coeficientes nulos, análogo ao que fizemos para o caso real. O algoritmo fica:

ALGORITMO 3.2.2 *Dados* $b = (0, 0, \ldots, 0, a_0, a_1, \ldots, a_{n-m}) \in \mathbb{R}^{n+1}$ *e* $A = 1$*;*

Para $k = 1, \ldots, m$

$c_k = 0$; $aux = -2(m-k)\alpha^{2(m-k)-1}$*;*

Para $j = 1, \ldots, 2(m-k)$

$c_k = c_k + aux$*;*

$aux = -\frac{aux}{\alpha}(\frac{2(m-k)+1}{j+1} - 1)$*;*

Fim

$c_k = Ac_k$*;*

$A = A\beta^2(\frac{m+1}{k+1} - 1)$*;*

Fim

Para $i = 2m+1, \ldots, 3m$

Para $j = 1, \ldots, i-(2m)$

$b_i = b_i - c_j b_{i-j}$;

Fim

Fim

Para $i = 3m+1, \ldots, n+1$

Para $j = 1, \ldots, m$

$b_i = b_i - c_j b_{i-j}$;

Fim

Fim

Fazendo a contagem de operações necessárias no algoritmo 3.2.2, para os dois primeiros laços, onde computamos os c_k, conseguimos para o laço em j $5(2m-k)$ somas e multiplicações. Somando esta quantia com as operações necessárias realizadas no exterior do laço em j, ficamos com $6(2m-k+1)$ multiplicações e $5(2m-k)+4$ somas.

Computando as operações necessárias para os dois laços duplos restantes, obtemos respectivamente $\sum_{i=2m+1}^{3m} i-2m$ e $\sum_{i=3m+1}^{n+1} m$ somas e multiplicações, totalizando $nm-\frac{5m^2}{2}+\frac{3m}{2}$. Para se ter uma ideia do custo desnecessário que o algoritmo 3.1.2 faria para se deflacionar no caso complexo, basta lembrar que a contagem para computar $b(z)$ sendo $d(z) = (z-\alpha)^m$ totalizava $nm - \frac{m^2}{2} - \frac{m}{2}$, o que já é maior que o custo calculado acima, que divide pelo fator de multiplicidade duplamente maior $[(z-\alpha+\beta i)(z-\alpha-\beta i)]^m$.

Isto prova que os algoritmos 3.1.3 e 3.2.2 são mais eficientes que os já existentes para se deflacionar polinômios, podendo ser usados para se diminuir o custo computacional de métodos para determinar raízes de polinômios.

4 UM MÉTODO PARA CÁLCULO DE RAÍZES DE POLINÔMIOS

4.1 Introdução

Neste capítulo apresentamos um método para cálculo de raízes de um polinômio de coeficientes reais da forma (3.1). O método é baseado na ideia do processo de deflação descrito no capítulo anterior. Embora possam haver generalizações do método aqui proposto para determinar raízes de multiplicidades maiores, consideraremos apenas os casos em que queremos calcular uma raiz simples ou uma raiz dupla. Considerações sobre possíveis generalizações serão feitas na seção 5.2, do capítulo 5.

O método aqui apresentado usa uma iteração de ponto fixo que converge para um valor que satisfaz a relação de recorrência descrita no teorema 3.1.1. As relações tornam-se relações conhecidas para os casos $m = 1$, e $m = 2$: Para $m = 1$, a relação é a do processo de deflação usual enquanto que para $m = 2$, é a relação utilizada no método de Bairstow, por exemplo. (veja [9]).

Embora o método de Bairstow use da relação de recorrência para obter um método iterativo para encontrar raízes, as ideias dos métodos são diferentes.

Uma vez demonstrado o teorema 3.1.1, a recíproca é um corolário óbvio do algoritmo de Euclides, logo, pode-se ver que α é uma raiz de (3.1) se e somente se

o polinômio quociente satisfaz as relações de recorrência descritas no teorema 3.1.1. Dito isso, podemos discutir o algoritmo, bem como expor alguns resultados sobre sua convergência.

Organizaremos da seguinte forma: na seção 4.2 apresentamos os dois algoritmos, e na seção 4.3 exibiremos provas de convergência de casos particulares, e algumas considerações sobre generalizações e estudos futuros serão feitas na seção 5.2 do capítulo 5.

4.2 Os Algoritmos

Primeiro tratamos do caso simples, onde dividimos o polinômio (3.1) pelo fator linear $(z-\alpha)$, e queremos que a divisão seja exata. Chamando o polinômio quociente de $B(z) = b_0 z^{n-1} + b_1 z^{n-2} + \cdots + b_{n-2} z + b_{n-1}$, sabemos, pelo teorema 3.1.1 que se a divisão é exata, os coeficientes de $B(z)$ satisfazem

$$b_j = a_j + \alpha b_{j-1}. \tag{4.1}$$

Então, tentamos raciocinar de trás para a frente, e construímos uma sequência que satisfaz (4.1), a cada passo. Mais precisamente, seja $B_0(z) = b_0^0 z^{n-1} + b_1^0 z^{n-2} + \cdots + b_{n-2}^0 z + b_{n-1}^0$, onde $b_j^0 = a_j + \alpha^0 b_{j-1}^0$, e α^0 é qualquer valor diferente de 0. Agora queremos encontrar um modo razoável de se obter α_1, e então fazer $B_1(z)$ análogo à $B_0(z)$, mas satisfazendo $b_j^1 = a_j + \alpha^1 b_{j-1}^1$, e assim por diante. Faremos isso da seguinte forma:

Tomando $p(z) = B(z)(z-\alpha)$, onde α é a raiz que queremos obter, e $p_0(z) = B_0(z)(z-\alpha_0)$, computando a diferença entre $p(z)$ e $p_0(z)$, conseguimos

$p(z) - p_0(z) =$

$$(a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n) - (b_0^0 z^{n-1} + b_1^0 z^{n-2} + \cdots + b_{n-2}^0 z + b_{n-1}^0)(z - \alpha_0) =$$

$$(a_0 - b_0^0)z^n + (a_1 - (b_1^0 - \alpha_0 b_0^0))z^{n-1} + \cdots + (a_j - (b_j^0 - \alpha_0 b_{j-1}^0))z^j + \cdots + (a_{n-1} - (b_{n-1}^0 - \alpha_0 b_{n-2}^0)))z + (a_n + \alpha_0 b_{n-1}^0)$$

Pela definição dos b_j^0, vemos que os primeiros $n-1$ termos da expressão acima se cancelam, e assim, temos que $p(z) - p_0(z) = a_n + \alpha_0 b_{n-1}^0$.

Finalmente, uma vez que desejamos que o resíduo $p(z) - p_0(z)$ fique próximo de zero (para então obtermos $p_0(z) \simeq p(z)$), fazemos $\phi_0(\alpha) = a_n - \alpha b_{n-1}^0$, e inspirando-nos no teorema do ponto fixo, tomamos α_1 como a raiz de $\phi_0(\alpha)$, o que nos dá $\alpha_1 = \frac{-a_n}{b_{n-1}^0}$, desde que $b_{n-1}^0 \neq 0$. E, mais geralmente, tomando $\{\alpha_{k+1}\}$ como a sequência dos zeros de $\phi_k(\alpha)$, obtemos a seguinte expressão para a sequência $\{\alpha_k\}$

$$\alpha_{k+1} = \frac{-a_n}{b_{n-1}^k}, \tag{4.2}$$

onde b_{n-1}^k é assumido não nulo e computado recursivamente, pela relação (4.1), mas com índices k. Agora estamos prontos para apresentar o algoritmo.

Dado um polinômio $p(z)$ da forma (3.1), desde que $a_0 a_n \neq 0$ e um valor inicial $\alpha \neq 0$, o algoritmo abaixo gera uma sequência que se convergir, converge para uma raiz real do polinômio. A convergência será considerada na seção 4.3:

ALGORITMO 4.2.1 *Dados a =* $(a_0, \ldots, a_n)^t \in \mathbb{R}^{n+1}$*, os coeficientes do polinômio,* α*,* $\epsilon > 0$*, e* $d > \epsilon$*,*

$k = 1$.

Enquanto $|d| \geq \epsilon$

$A = a_0$*;*

Para $j = 0, \ldots, n-1$

$A = \alpha A + a_{j+1}$*;*

Fim

$d1 = a_n + \alpha A$*;*

$d =$*Min*$(|d1|, |A|)$

$\alpha = \frac{-a_n}{A}$*;*

$k = k + 1$*;*

Fim

Note que ambos os laços custam $O(n)$ operações, e logo cada iteração do algoritmo é $O(n)$. O primeiro laço calcula a relação (4.1), e ao mesmo tempo é uma avaliação de Horner de $p(\alpha_k)$, temos dois critérios de parada, A é a avaliação de Horner de $p(\alpha_k)$, e d_1, o módulo da diferença $p(z) - p_k(z)$. Note que se $p(z) - p_k(z) = 0$, temos que α_k é uma raiz de $p(z)$. Finalmente, computamos o "novo" α fazendo $\alpha = \frac{-a_n}{A}$, que é a iteração de ponto fixo descrita acima. Mostraremos na próxima seção que esta iteração converge.

Podemos deduzir outro algoritmo que calcula uma raiz dupla de $p(z)$ seguindo o mesmo processo, mas ao invés de dividir o polinômio pelo fator $(z - \alpha)$, o dividimos por $(z - \alpha)^2$, e obtemos o polinômio quociente $B(z)$ de grau $n - 2$. Os coeficientes b_j^i

são obtidos pelo teorema 3.1.1 e possuem a forma

$$b_j^i = a_j + 2\alpha_i b_{j-1}^i - \alpha_i^2 b_{j-2}^i, \;\; j = 0, 1, \ldots, n-2, \tag{4.3}$$

com $b_{-1} = b_{-2} = 0$.

Então, basicamente o que precisa ser modificado é a sequência dos b_i e a fórmula para a atualização de α no fim do laço.

Construíndo os polinômios $B_k(z)$ e $p_k(z)$ analogamente ao que foi feito para o caso simples, vemos que a diferença $R_k(z) = p(z) - p_k(z)$ é:

$R_k(z) =$

$a_0 z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n - (z^2 - 2\alpha_k z + \alpha_k^2)(b_0^k z^{n-2} + b_1^k z^{n-3} + \ldots + b_{n-3}^k z + b_{n-2}^k) =$

$a_0 z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n -$

$-(b_0^k z^n + (b_1^k - 2\alpha_k b_0^k) z^{n-1} + (b_2^k - 2\alpha_k b_1^k + \alpha_k^2 b_0^k) z^{n-2} + \ldots + (-2\alpha_k b_{n-2}^k + \alpha_k^2 b_{n-3}^k) z + \alpha_k^2 b_{n-2}^k).$

E usando a fórmula de recorrência dos b_j, temos que

$$R_k(z) = (a_{n-1} + 2\alpha_k b_{n-2}^k - \alpha_k^2 b_{n-3}^k) z + (a_n - \alpha_k^2 b_{n-2}^k). \tag{4.4}$$

Denotando $C_k = a_n - \alpha_k^2 b_{n-2}^k$ e $b_{n-1}^k = a_{n-1} + 2\alpha_k b_{n-2}^k - \alpha_k^2 b_{n-3}^k$, podemos escrever $R_k(z) = b_{n-1}^k z + C_k$.

No nosso algoritmo principal, a ser apresentado posteriormente, a sequência gerada começa com α_0 dado, e determina α_{k+1} de modo que $R_k(z) = 0$, ou seja

$$\alpha_{k+1} = -\frac{C_k}{b_{n-1}^k}, \tag{4.5}$$

com $b_{n-1}^k \neq 0$. A necessidade deste número ser não nulo para cada valor de k será

considerada posteriormente. A próxima proposição será útil para decidir se a raiz é simples ou não.

PROPOSIÇÃO 4.2.1 *Seja $z = \alpha$ uma raiz do polinômio (3.1) e $p(z) = B(z)(z-\alpha)^2+R(z)$ com $R(z) = bz + c$ sendo o grau de $B(z)$ igual à $n - 2$. Se $b = 0$ então α é uma raiz dupla, caso contrário é simples.*

Demonstração. Primeiro escrevemos a expansão de Taylor de $p(z)$ em torno de α:

$$p(z) = \sum_{j=0}^{n} \frac{p^{(j)}(\alpha)}{j!}(z-\alpha)^j.$$

Que naturalmente é finita pois $p(z)$ é função polinomial. Escrevendo por extenso ficamos com

$$p(z) = \frac{p^{(n)}(\alpha)}{n!}(z-\alpha)^n + \cdots + p'(\alpha)(z-\alpha) + p(\alpha).$$

Como $p(\alpha) = 0$, podemos colocar o fator quadrático $(z-\alpha)^2$ em evidência em todos os termos exceto $p'(\alpha)(z-\alpha)$, e obter

$$p(z) = B(z)(z-\alpha)^2 + p'(\alpha)(z-\alpha),$$

e comparando com a fórmula do enunciado, temos que $p'(\alpha)(z-\alpha) = p'(\alpha)z-\alpha p'(\alpha) = R(z)$, com b do enunciado sendo exatamente $p'(\alpha)$. Claramente se $p'(\alpha) = 0$, $(z-\alpha)^2$ divide $p(z)$, e a raíz é dupla. Por outro lado, se $(z-\alpha)^2|p(z)$ então existe $g(z)$ com

$$g(z)(z-\alpha)^2 = p(z) = B(z)(z-\alpha)^2 + p'(\alpha)(z-\alpha).$$

Considerando esta última igualdade, subtraindo $B(z)(z-\alpha)$, e dividindo ambos termos por $(z-\alpha)$, chegamos à:

$$(g(z) - B(z))(z-\alpha) = p'(\alpha).$$

Como a igualdade vale para todo z, em particular pondo $z = \alpha$ no primeiro termo, segue que $p'(\alpha) = 0$, completando o resultado.

■

É importante enfatizar que o algoritmo detecta se a raiz é simples ou não. Embora estejamos interessados em raízes duplas, o algoritmo também determina as raízes simples.

O algoritmo começa com um valor inicial α, um polinômio $p(z)$ da forma (3.1) e gera uma sequência que converge para uma raiz de $p(z)$, usando a fórmula (4.5).

ALGORITMO 4.2.2 *Dados a =* $(a_0, \ldots, a_n)^t \in \mathbb{R}^{n+1}$*, os coeficientes do polinômio,* α*,* $\epsilon > 0$ *e true=1*

$k = 1$

Enquanto true=1

$b_0 = a_0$, $b_1 = a_1 + 2\alpha a_0$;

Para $j = 2, \ldots, n-1$

$b_j = a_j + 2\alpha b_{j-1} - \alpha^2 b_{j-2}$;

Fim

$d = a_0$;

Para $i = 0, \ldots, n-1$

$d = \alpha d + a_{i+1}$;

Fim

Se $|d| \leq \epsilon$

true=0

senão

$$\alpha = \frac{a_n - \alpha^2 b_{n-2}}{b_{n-1}};$$

Fim

$k = k + 1$;

Fim

se $b_{n-1} = 0$

α *é uma raiz dupla.*

senão

α *é uma raiz simples.*

Fim

A seguir justificaremos os detalhes do algoritmo. Embora as relações (4.4) e (4.5) usem apenas as três últimas entradas de b nós salvamos todos os coeficientes. Se a raiz determinada é dupla, então estes coeficientes coincidem com os do polinômio deflacionado de grau $n - 2$. Isto economiza várias operações, no caso de se aplicar o algoritmo diversas vezes para encontrar todas as raízes.

O primeiro laço interno (em j) determina os coeficientes b_j e o segundo (em i) é aplicação do processo de Horner para avaliar $p(\alpha)$ como critério de parada. Enfatizamos aqui que outros critérios de parada podem ser usados, de modo que este laço não é estritamente necessário. uma vez que $d = p(\alpha)$ e se $|d| \leq \epsilon$ então α é uma aproximação para a raiz com precisão ϵ e o algoritmo pára.

Finalmente checamos se $b_{n-1} = 0$ (Lembre que este é o coeficiente de $R_k(z)$

na relação (4.4)). Se este coeficiente é zero segue pela proposição 4.2.1 que a raiz é dupla.

Contudo, se a raiz é simples, então a deflação deve ser feita dividindo-se $p(z)$ pelo fator $(z-\alpha)$, que não nos fornece os mesmos $B(z)$ armazenados. mas a deflação neste caso custa apenas $O(n)$ operações e pode ser feita usando-se a relação (4.1).

4.3 Análise de Convergência

Nesta seção mostraremos que as sequências geradas por (4.2) e (4.5) convergem. Começaremos pela primeira. Embora estejamos gerando uma sequência baseando-nos no teorema do ponto fixo, não podemos usar este para provar a convergência, pois a função $z \mapsto p(z)$ pode não satisfazer $p(\mathbb{R}) = \mathbb{R}$, o que é uma das hipóteses necessárias para o teorema do ponto fixo. Logo, devemos provar a convergência de outra maneira. Começamos assumindo certas hipóteses que valem quando aplicamos o algoritmo para exemplos concretos. Tais hipóteses são

$$\liminf|\alpha_k| \neq 0, \quad \liminf|p(\alpha_k) - a_n| \neq 0 \tag{4.6}$$

Assumindo isso, concluímos não somente que $\alpha_k \neq 0$ para todo k, mas também que α_k não possui uma subsequência que converge para 0 em módulo. Desta forma, vemos que a razão na relação (4.2) está definida para todo k, pois

$$b_{n-1}^k = a_{n-1} + \alpha_k b_{n-2}^{k-1},$$

e usando a relação de recorrência (4.1),

$$b_{n-1}^k = a_{n-1} + \alpha_k(a_{n-2} + \alpha_k(\cdots + \alpha_k a_0)\dots),$$

ou seja

$$b_{n-1}^{k} = \sum_{j=0}^{n-1} a_j \alpha_k^{n-j} = (p(\alpha_k) - a_n)/\alpha_k. \tag{4.7}$$

Isto implica que a razão só é zero (ou computacionalmente zero) ou indefinida se uma das equações não permitidas em (4.6) valerem. É razoável admitir que a_n e 0 não pertencem ao conjunto de pontos de acumulação de $\{\alpha_k\}$, uma vez que nem um nem outro são raízes de $p(z)$ por hipótese. Então escolhendo um $\alpha_0 \in \mathbb{R}$ não nulo, temos o seguinte resultado.

LEMA 4.3.1 *Se $\alpha_0 \in \mathbb{R} - \{0\}$, então a sequência (4.2) possui um ponto de acumulação.*

Demonstração. De fato, como o conjunto $\mathbb{R}$ é fechado para as operações em (4.2), e estas estão sempre definidas, pela observação acima, vemos que $\alpha_k \in \mathbb{R}$, para todo k. Também, pela segunda hipótese (4.6), e pela relação (4.7), existe $\epsilon > 0$ tal que $|b_{n-1}^{k}| > \epsilon$ para todo k, logo, pela fórmula 4.2, a sequência $\{\alpha_k\}$ é limitada. E portanto, pelo teorema de Bolzano-Weirstrass, tem uma subsequência convergente e logo $\{\alpha_k\}$ tem ponto de acumulação. ∎

No lema 4.3.1 provamos que o conjunto de pontos de acumulação de $\{\alpha_k\}$ é não vazio. No nosso próximo resultado provamos que a função de iteração $f(x) = -\frac{a_n}{b_{n-1}(x)}$ é Lipschitziana.

TEOREMA 4.3.2 *A iteração dada por (4.2) tem uma subsequência que converge para uma raiz de $p(z)$.*

Demonstração. Com efeito, para todo x e y, temos que

$$|f(x) - f(y)| = |\frac{a_n}{b_{n-1}(y)} - \frac{a_n}{b_{n-1}(x)}| = |\frac{a_n(b_{n-1}(x) - b_{n-1}(y))}{b_{n-1}(y)b_{n-1}(x)}|.$$

Da hipótese (4.6), sabemos que existe uma constante δ limitando b_{n-1} inferiormente, logo

$$|f(x) - f(y)| \leq |\frac{a_n(b_{n-1}(x) - b_{n-1}(y))}{\delta^2}|.$$

Para $x = y$ a inequação da definição de função Lipschitziana é satisfeita para qualquer constante, tomemos sem perda de generalidade $x \neq y$. Como $b_{n-1}(x)$ é um polinômio de grau $n-1$ em x, dividindo-o por $x-y$, obtemos que $\frac{b_{n-1}(x)}{x-y} = D(x,y) + \frac{R}{x-y}$, onde $D(x,y)$ é um polinômio de grau $n-2$, e R uma constante dependendo de y. O análogo acontece dividindo-se $b_{n-1}(y)$ por $x-y$. Assumindo que $\sup\{x; x \in Dom(f)\}$ e $\sup\{y; y \in Dom(f)\}$ são finitos, conseguimos uma constante $C > 0$ limitando $D(x,y)$. Multiplicando e dividindo a inequação por $x - y$, temos

$$|f(x) - f(y)| \leq |\frac{a_n(b_{n-1}(x) - b_{n-1}(y))(x-y)}{\delta^2|x-y|}|.$$

Aplicando a fórmula acima para $\frac{b_{n-1}(.)}{x-y}$, limitando $D(x,y)$ por C, e percebendo que, $\frac{|R(y)-R(x)|}{|x-y|}$ se limita por uma constante D, obtemos

$$|f(x) - f(y)| \leq \frac{(C+D)a_n}{\delta^2}|x-y|,$$

para quaisquer $x \neq y$, donde segue que f é Lipschitziana. Como a sequência α_k tem um ponto de acumulação e a função de iteração é Lipschitziana, temos, à luz da proposição 2.1.2, que f converge para um ponto fixo quando α_{k_j} converge para um ponto de acumulação, e por construção, o algoritmo então converge nesta mesma subsequência. ∎

Note que aqui assumimos a existência de $\sup_{x,y}\{x, y \in Dom(f)\}$, o que é basicamente supor que f está definida numa bola limitada. Não se perde generalidade com isso, pois já havíamos mostrado que α_k está contida numa bola de $\mathbb{R}$.

Estes resultados asseguram que o algoritmo 3.1.1 certamente determinará uma aproximação para a raiz real de $p(z)$, dada certa tolerância.

Para o caso da raiz dupla, procederemos com argumento similar. Primeiro mostraremos que a sequência real obtida por (4.5) é limitada, e então usaremos o argumento da função Lipschitziana novamente para provar que existe um ponto de acumulação o qual é raiz de $p(z)$. Aqui precisaremos da seguinte generalização da hipótese (4.6):

$$\liminf|\alpha_k| > 0, \quad \liminf|b_{n-2}^k| > 0 \quad e$$
$$\liminf|a_{n-1} + 2\alpha_k b_{n-2}^k - \alpha_k^2 b_{n-3}^k| > 0 \tag{4.8}$$

Então escolhendo $\delta > 0$ tal que $|b_{n-1}^k| > 0$ para todo k, nós assumiremos que $|\alpha_k| \leq M$ para uma constante positiva M e mostraremos que $|\alpha_{k+1}| \leq M$ (o mesmo M). Desta forma a sequência é limitada superiormente por M e limitada inferiormente por outra constante, pela hipótese (4.8). Note que provar que $|\alpha_{k+1}| \leq M$ é o mesmo que provar que $|\frac{\alpha_k^2 b_{n-2}^k - a_n}{b_{n-1}^k}| \leq M$ o que implica, já que $|b_{n-1}^k| > \delta$, em mostrar que $|\alpha_k^2 b_{n-2}^k - a_n| \leq M\delta$. Desta forma nos concentraremos em provar esta última inequação, e o faremos fazendo o uso de dois lemas: o primeiro nos permite encontrar uma expressão para os b_j, e o segundo mostra a limitação da sequência $\{\alpha_k\}$.

LEMA 4.3.3 *Para todo $k \in \mathbb{N}$, os coeficientes de $B_k(z) = b_0^k z^{n-1} + b_1^k z^{n-2} + \cdots + b_{n-3}^k z + b_{n-2}^k$ satisfazem*

$$b_j^k = H(a_{0,\dots,j},(2\alpha_k)) + \sum_{i=1}^{Floor(\frac{j}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-2i} (\alpha_k)^{j-2i-l} C_l H(a_{0,\dots,l}, 2\alpha_k) \right], \tag{4.9}$$

onde $H(a_{0,\dots,s}, x)$ *é a avaliação de Horner em* x *do polinômio dado pelos coeficientes* $a_{0\dots s}$*. Isto é,* $H(a_{0,\dots,s}, x) = a_s + x(a_{s-1} + x(\cdots + a_1(x + a_0))\dots)$ *e* C_l *são constantes tais que* $|C_l| < 3^j$*.*

Demonstração. Para $j = 0$, temos que $b_0^k = a_0$ para todo k, e uma vez que $H(a_{0,\dots,0}, 2\alpha_k) = a_0$, obtemos a tese automaticamente. Suponha agora que a igualdade seja verdadeira para todos valores de índices até $j-1$, provaremos que também vale para b_j^k (lembramos que k está fixado, logo não desempenha papel importante na demonstração).

Usando a relação (4.3) e a hipótese de indução, temos que

$$b_j^k = a_j + 2\alpha_k H(a_{0,\dots,j-1}, 2\alpha_k) + 2\alpha_k \sum_{i=1}^{Floor(\frac{j-1}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-1-2i} C_l \alpha_k^{j-1-2i-l} H(a_{0,\dots,l}, 2\alpha_k) \right] -$$

$$-\alpha_k^2 \left[H(a_{0,\dots,j-2}, 2\alpha_k) + \sum_{i=1}^{Floor(\frac{j-2}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-2-2i} C_l \alpha_k^{j-2-2i-l} H(a_{0,\dots,l}, 2\alpha_k) \right] \right].$$

Rearranjando os termos convenientemente, obtemos

$$b_j^k = H(a_{0,\dots,j}, 2\alpha_k) + \sum_{i=1}^{Floor(\frac{j-1}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-1-2i} C_l \alpha_k^{j-2i-l} H(a_{0,\dots,l}, 2\alpha_k) \right] -$$

$$-\alpha_k^2 H(a_{0,\dots,j-2}, 2\alpha_k) + \sum_{i=2}^{Floor(\frac{j}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-2i} C_l \alpha_k^{j-2i-l} H(a_{0,\dots,l}, 2\alpha_k) \right].$$

Aqui usamos que $H(a_{0,\dots,j}, 2\alpha_k) = 2\alpha_k H(a_{0,\dots,j-1}, 2\alpha_k)$ para "retirar" o $2\alpha_k$ multiplicando o primeiro termo, e para os outros termos, colocamos o $2\alpha_k$, bem como o

α_k^2 dentro do somatório, e alteramos os índices de modo a não modificar a expressão anterior.

Finalmente, agrupando os termos de índices comuns, colocando todos em evidência, e renomeando as somas resultantes de C_l para $\overline{C}_l$, temos que

$$b_j^k = H(a_{0,\ldots,j}, 2\alpha_k) + \sum_{i=1}^{Floor(\frac{j}{2})} (-\alpha_k)^{2i} \left[\sum_{l=0}^{j-2i} \overline{C}_l \alpha_k^{j-2i-l} H(a_{0,\ldots,l}, 2\alpha_k) \right].$$

Como estamos supondo $|C_l| < 3^{j-1}$, e notando que $\overline{C}_l$ não excede $3C_l$, pois para cada índice l as constantes que acompanham os termos de cada somatório são no máximo um C_l, podendo não ser nenhum, e o primeiro termo da segunda linha contribui com uma constante para apenas um dos índices l, temos assim que $|\overline{C}_l| < 3^j$, e a demonstração se dá por indução.

■

LEMA 4.3.4 *Considere a hipótese (4.8) e a sequência*

$$\alpha_{k+1} = -\frac{a_n - \alpha_k^2 b_{n-2}^k}{a_{n-1} + 2\alpha_k b_{n-2}^k - \alpha_k^2 b_{n-3}^k}. \tag{4.10}$$

Suponha ainda que $|a_n| < \delta$*, onde* δ *é uma constante tal que* $|b_{n-1}^k| > \delta$ *para todo* k*. Então existe* $M > 1$ *tal que* $|\alpha_0| < M$ *implica que para todo* $k \in \mathbb{N}$, $|\alpha_k| < M$*.*

Demonstração.

já vimos anteriormente que basta provar que $|\alpha_k^2 b_{n-2}^k - a_n| \leq M\delta$ para algum $M \geq 1$. Note que a demonstração é recursiva, isto é, assumiremos que $\alpha_k \leq M$ com $M > 1$, e ao resolver a inequação acima, isto nos dará vários valores de M

que também limitam α_{k+1}. Mas não é suficiente tomar qualquer um destes valores encontrados, precisamos ainda mostrar que a solução é maior que 1, ou obteríamos uma contradição. Expandindo a expressão $a_n - \alpha_k^2 b_{n-2}^k$ em b_{n-2}^k, e usando a relação (4.9), temos que

$$a_n - \alpha_k^2(H(a_{0,\ldots,n-2},(2\alpha_k)) + \sum_{i=1}^{Floor(\frac{n-2}{2})}(-\alpha_k)^{2i}\left[\sum_{l=0}^{n-2(i+1)}(\alpha_k)^{n-2(i+1)-l}C_l H(a_{0\ldots l},2\alpha_k)\right].$$

Como $H(a_{0,\ldots,n-2},2\alpha_k) = \sum_{i=0}^{n-2} a_i(2\alpha_k^{n-2-i})$ e $M \geq 1$ podemos limitar a avaliação de Horner por $(n-2)2M^{n-2}A$, onde $A = \max_i\{|a_i|\}$ e C é uma constante arbitrária que limita C_l superiormente para todo l.

Assim, limitando α_k por M, obtemos que

$$|\alpha_k^2 b_{n-2}^k - a_n| \leq |\alpha_k^2 b_{n-2}^k| + |a_n| \leq$$

$$\leq |M^2(n-2)2M^{n-2}A + \frac{n-2}{2}M^{n-2}(n-2)^2M^{n-2}C2M^{n-2}A| + |a_n| \leq$$

$$\leq M|a_n| + 2M^{2n+1}(n-2)A + M^{4n+1}(n-2)^3AC.$$

Como queremos que esta grandeza seja menor ou igual à $M\delta$, e $M \geq 1$ temos, dividindo ambos lados por M, que isto é equivalente a inequação quadrática

$$(|a_n| - \delta) + 2M^{2n}(n-2)A + M^{4n}(n-2)^3CA \leq 0.$$

Resolvendo para M^{2n}, obtemos

$$M^{2n} = -\frac{2(n-2)A \pm \sqrt{(2(n-2)A)^2 - 4(|a_n|-\delta)(n-2)^3AC}}{2((n-2)^C A)}.$$

Como o discriminante é $\Delta = 4[(n-2)^2A^2 - (n-2)^3AC(|a_n|-\delta)] = 4(n-$

$2)^2A(A-(n-2)C(|a_n|-\delta))$, as raízes são reais.

Ainda, como supomos que $|a_n| < \delta$, escolhendo o sinal negativo conseguimos uma raiz positiva, e explorando a liberdade de escolha de $|a_n|$, iremos agora mostrar que podemos tomar os coeficientes a_i de modo que M seja de fato maior que 1.

Precisamos mostrar que a inequação

$$-\frac{2(n-2)A-\sqrt{(2(n-2)A)^2-4(|a_n|-\delta)(n-2)^3AC}}{2((n-2)^2CA)} > 1$$

tem solução para C, desde que este limite as constantes C_l superiormente. Isto é equivalente à:

$$\sqrt{(2(n-2)A)^2-4(|a_n|-\delta)(n-2)^3AC} > 2(n-2)^2AC+2(n-2)A.$$

elevando os dois lados ao quadrado (observando que ambos são positivos) temos a inequação equivalente

$$(2(n-2)A)^2-4(|a_n|-\delta)(n-2)^3AC > 4(n-2)^2A^2[(n-2)^2C^2+2C(n-2)+1],$$

ou ainda,

$$4(n-2)^2A[A-(n-2)(|a_n|-\delta)C] > 4(n-2)^2A[(n-2)^2C^2A+2AC(n-2)+A].$$

dividindo os dois lados pela grandeza positiva $4(n-2)^2A$, obtemos a inequação

$$(n-2)^2C^2A+2AC(n-2)+(n-2)(|a_n|-\delta)C < 0.$$

resolvendo para A, vemos que este deve satisfazer

$$A < \frac{\delta-|a_n|}{(n-2)C+2}$$

e finalmente, escolhendo A pequeno suficiente para que a inequação valha, vemos

que existe C satisfazendo a relação, e logo a inequação tem solução para M, completando a prova.

■

Note que assumir que $|a_n| < \delta$, bem como escolher A suficientemente pequeno não acarreta em perda de generalidade, pois podemos escalonar o polinômio dado, uma vez que não estamos exigindo que este seja mônico. Embora tenhamos precisado desta hipótese adicional para demonstrar o lema, o algoritmo funciona corretamente sem efetuar o escalonamento.

Uma vez mostrado que existe M limitando a sequência α_k superiormente, pode-se mostrar que a função de iteração

$$f(x) = -\frac{a_n - x^2 b_{n-2}(x)}{b_{n-1}}$$

é Lipschitziana. Sendo $b_{n-2}(x)$ como no lema 4.3.3. De fato, para quaisquer x e y, temos que

$$\begin{aligned} |f(x) - f(y)| &= \left| \frac{a_n - y^2 b_{n-2}(y)}{b_{n-1}(y)} - \frac{a_n - x^2 b_{n-2}(x)}{b_{n-1}(x)} \right| = \\ &\left| \frac{(a_n - y^2 b_{n-2}(y)) b_{n-1} x - (a_n - x^2 b_{n-2}(x)) b_{n-1}(y)}{b_{n-2}(y) b_{n-1}(x)} \right| \leq \\ &\left| \frac{x^2 b_{n-2}(x) b_{n-1}(y) - y^2 b_{n-2}(y) b_{n-1}(x)}{\delta^2} \right|. \end{aligned} \tag{4.11}$$

esta última desigualdade é satisfeita pela hipótese 4.8, porque existe δ que limita a função b_{n-2} inferiormente, independentemente de x e y.

Como α_k é limitada superiormente, também o são b_{n-1}^k e b_{n-2}^k, e portanto, podemos limitar $|f(x) - f(y)|$ por $\frac{M}{\delta^2}|y^2 - x^2|$.

Finalmente, percebendo que $|y^2 - x^2| = |(x-y)(x+y)| \leq |(x-y)|(|x| + |y|)$, e

limitando x e y por uma certa constante N (o que é basicamente restringir o domínio de f à uma bola, que não acarreta em perda de generalidade, pois estamos assumindo que a sequência $\{\alpha_k\}$ é limitada superiormente), obtemos que

$$|f(x) - f(y)| \leq \frac{2MN}{\delta^2}|x - y|,$$

e portanto a função iteração é Lipschitziana.

usando a versão complexa do teorema de Bolzano-Weirstrass, vemos que a sequência $\{\alpha_k\}$ neste caso também tem ponto de acumulação, e pelo mesmo argumento feito no caso simples, temos que a função iteração converge para um ponto fixo nesta mesma subsequência, assegurando a convergência do algoritmo.

5 EXPERIMENTOS NUMÉRICOS E CONCLUSÃO

5.1 Testes Numéricos

Nesta Seção apresentaremos testes numéricos para ilustrar o desempenho dos algoritmos propostos. Começaremos mostrando resultados dos algoritmos 3.1.3 e 3.2.2, e em seguida exporemos os resultados à respeito dos métodos 4.2.1 e 4.2.2.

5.1.1 Testes Numéricos dos Algoritmos de Deflação

Aqui exporemos os resultados obtidos ao se aplicar os algoritmos 3.1.3 e 3.2.2 à alguns polinômios dados a seguir. Usamos o algoritmo 3.2.2 sempre e somente quando as raízes em questão eram complexas conjugadas.

Exemplo 1: $p(z) = (z-2)^4(z-1)$.

Se o algoritmo é usado para a raiz $\alpha = 2$, o polinômio resultante é $p(z) = z-1$.

Por outro lado, se a raiz é $\alpha = 1$ o polinômio resultante é $p(z) = z^4 - 8z^3 + 24z^2 - 32z + 16$.

Exemplo 2: $p(z) = (z-0.1)^4(z-0.2)^3(z-0.3)^2(z-0.4)$.

Usando o algoritmo para a raiz $\alpha = 0.1$ o polinômio resultante é $p(z) = 0.000288 - 0.00696z + 0.0692z^2 - 0.362z^3 + 1.05z^4 - 1.6z^5 + z^6$.

Para a raiz $\alpha = 0.2$ temos $p(z) = -0.0000036+0.000177z-0.00358z^2+0.0383z^3-0.232z^4+0.79z^5-1.4z^6+z^7$.

Para $\alpha = 0.3$ obtemos $p(z) = 0.0000003-0.0000184z+0.000452z^2-0.00618z^3+0.0513z^4-0.264z^5+0.82z^6-1.4z^7+z^8$.

E por fim, para $\alpha = 0.4$ vem $p(z) = -4.000D-08+0.0000044z-0.000119z^2+0.001817z^3-0.0174z^4+0.1083z^5-0.438z^6+1.11z^7-1.6z^8+z^9$.

Exemplo 3: $p(z) = (z-0.001)(z-0.01)(z-0.1)(z-(0.1+0.0000001i))((z-(0.1-0.0000001i))(z-1)(z-10)$.

Usando o algoritmo para a raíz complexa $\alpha = 0.1+0.0000001i$ obtemos o polinômio $p(z) = -0.0007489+0.2822105z-3.2120461z^2+13.22839z^3-11.291z^4+z^5$.

Para os demais casos, o algoritmo também se comportou muito bem.

Exemplo 4: $p(z) = (z-1)^5$.

Usando o algoritmo com a raiz $\alpha = 1$ conseguimos $p(z) = 1$.

Exemplo 5: $p(z) = (z-i)^2(z+i)^2(z-(0.01+i))(z+(0.01-i))(z-(1.0000001i))(z+(1.0000001i)(z-3)$

Com a raiz complexa $\alpha = i$ de multiplicidade 2 o algoritmo retorna $p(z) = -3.0003+1.0601z-6.0203z^2+2.0601z^3-3.02z^4+z^5$.

5.1.2 Testes Numéricos do Algoritmo para Determinação de Raízes

Nesta seção apresentaremos alguns testes numéricos para mostrar o desempenho do algoritmo 4.2.2 com polinômios que aparecem em artigos da literatura. Embora tenhamos realizado também testes com o algoritmo 4.2.1 mostraremos apenas

os resultados do algoritmo 4.2.2 uma vez que este também determina as raízes simples. Foram realizados dois tipos de testes. Os primeiros testes visaram mostrar que o algoritmo encontra a raiz quando o ponto inicial é real ou complexo. O segundo conjunto de testes envolvem bacias de atração recentemente usadas para comparar a eficiência do algoritmo [4, 5, 7]. As bacias de atração são um modo de visualizar como o algoritmo se comporta como função de vários valores iniciais. Aqui usamos as bacias de atração para comparar a eficiência do algoritmo proposto quando este é comparado com os métodos de Newton e de Halley. Uma vez que em [4, 5] e [7] estes métodos mostraram um bom comportamento na maioria dos problemas testados. Usaremos estes métodos para comparar com o nosso. Os processos iterativos dos métodos de Newton e Halley para raízes múltiplas [5] são dados, respectivamente, por

$$x_{k+1} = x_k - m\frac{P_k}{P'_k} \tag{5.1}$$

e

$$x_{k+1} = x_k - \frac{P_k}{\frac{m+1}{2m}P'_k - \frac{P_k P''_k}{2P'_k}} \tag{5.2}$$

onde m é a multiplicidade da raiz e $P_k = P(x_k)$ e analogamente para P'_k e P''_k.

O método 5.1 é uma modificação do método de Newton para quando se conhece a multiplicidade m da raiz. O método 5.2 também têm sua dedução proveniente do método de Newton: Considere a função

$$g(x) = \frac{f(x)}{\sqrt{(|f'(x)|)}}.$$

Então cada raiz de f que não é raiz de sua derivada é raiz de g, e ao mesmo tempo cada raiz de g é uma raiz de f. Assim, aplicando o método de Newton em sua fórmula

clássica para g, obtemos:

$$x^{k+1} = x^k - \frac{g(x^k)}{g'(x^k)},$$

com $g'(x) = \frac{2[f'(x)]^2 - f(x)f''(x)}{2f'(x)\sqrt{|f'(x)|}}$. Substituíndo na equação acima obtém-se o método de Halley clássico. O método 5.2 é uma modificação deste para quando a multiplicidade é conhecida. A dedução deste, bem como análise de convergência e abordagem mais completa pode ser encontrada em [2], ou [6].

Embora tenhamos feito comparações usando bacias de atração não comparamos número de iterações. Este não é um bom meio de comparação pois o método proposto não faz uso de cálculo de derivadas enquanto os de Newton e Halley o fazem.

A seguir apresentamos os polinômios testados e seus resultados. Os polinômios de P_1 à P_4 são apresentados em [5] e os de P_5 à P_7 em [2]. O polinômio P_8 é uma adaptação do polinômio de Wilkinson tal que tem 1 como raiz múltipla.

Polinômio $P_1(z) = (z-1)^2(z+1)^2$ com raízes $z = \pm 1$ real e de multiplicidade 2.

Polinômio $P_2(z) = (z^3-1)^2$. Este Polinômio tem raízes 1, $(-1)^{2/3}$ e $-(-1)^{1/3}$. Todas as raízes duplas.

Polinômio $P_3(z) = (z^5-1)^3$ com raízes $-(-1)^{1/5}$, $(-1)^{2/5}$, $-(-1)^{3/5}$, $(-1)^{4/5}$ e 1. Todas de multiplicidade três.

Polinômio $P_4(z) = (z^7-1)^4$ com raízes $-(-1)^{1/7}$, $(-1)^{2/7}$, $-(-1)^{3/7}$, $(-1)^{4/7}$, $-(-1)^{5/7}$, $(-1)^{6/7}$ e 1. Todas de multiplicidade quatro.

Polinômio $P_5(z) = z^5 - 8z^4 + 24z^3 - 34z^2 + 23z - 6$ com raiz $z = 1$ tripla, $z = 2$ e $z = 3$ simples.

pol	α_0	it	raízes	α_0	it	raízes	α_0	it	raízes
P_1	2	21	1	3+i	23	1	-4-2i	24	1
P_2	2	23	1	3+i	25	1	-1+i	21	-0.5+0.8660i
P_3	2	31	1	3+i	38	1	0+i	22	0.3090+0.9510i
P_4	2	42	1	3+i	54	1	0+i	24	-0.2223+0.9747i
P_5	1.5	19	1	1.5+i	22	1	5+i	12	3
P_6	1.5	21	1	1.5+i	22	1	-5+i	11	-2.3333+0.9428i
P_7	1.5	21	1	1.5+i	24	1	-2+i	11	-1
P_8	1.5	28	1	1.5+i	9	2	2+i	9	2
P_8	0.5	23	1	0.5+i	24	1	3+i	8	3
P_8	-10	35	1	-10+i	35	1	7+i	14	6

TABELA 1: Resultados do algoritmo 4.2.2 para os oito polinômios apresentados acima

Polinômio $P_6(z) = 3z^4 + 8z^3 - 6z^2 - 24z + 19$ cuja raiz $z = 1$ é dupla e $z = -2.3333 \pm 0.9428i$ são raízes complexas.

Polinômio $P_7(z) = z^5 - 3z^4 + 2z^3 + 2z^2 - 3z + 1$ cuja raiz $z = 1$ tem multiplicidade 4 e $z = -1$ é simples.

Polinômio $P_8(z) = z^7 - 22z^6 + 196z^5 - 910z^4 + 2359z^3 - 3388z^2 + 2484z - 720$ cuja raiz $z = 1$ tem multiplicidade 2 e $z = 2, 3, 4, 5$ e 6 são simples.

Na tabela 1 mostramos os resultados do algoritmo 4.2.2. O algoritmo pára quando atinge a precisão $1e^{-10}$. A primeira coluna é o problema a ser resolvido. A segunda é o valor inicial, um número real. A terceira coluna é o número de iterações e a quarta a raiz encontrada. Da quinta à sétima colunas temos o mesmo que as primeiras mas com valor inicial complexo para a raiz. Nas três últimas colunas, repetimos o mesmo com diferentes valores iniciais complexos. repetimos os testes para o Polinômio P_8 para diferentes valores iniciais pois este é um Polinômio instável.

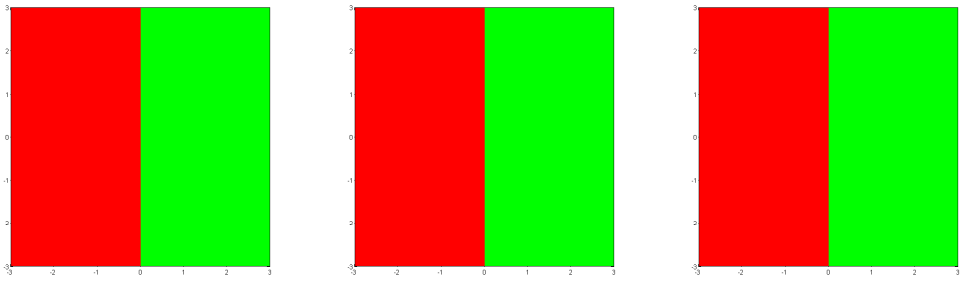
Note que quando usamos valores iniciais distintos o algoritmo encontra diferentes raízes como mostra a última coluna da tabela 1.

O método de Newton (5.1) não converge para o Polinômio P_8 para vários pon-

tos iniciais, por exemplo, $x_0 = 1.69; 1.7; 1.9$ entre outros. Isto também ocorre com os Polinômios P_6 e P_7. por isso fixamos 1000 como o número máximo de iterações para todos os pontos iniciais nos testes de bacias de atração.

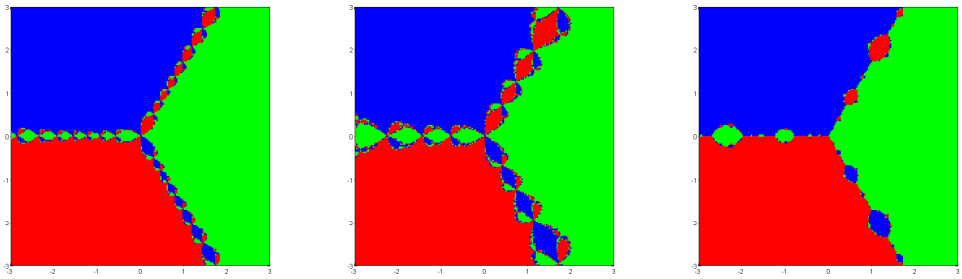
Mostramos os resultados das bacias de atração nas figuras 2 à 9 e em todas elas, a parte (a) diz respeito ao algoritmo proposto neste trabalho, enquanto (b) e (c) são, respectivamente, as bacias de Newton (5.1) e Halley (5.2). Para o polinômio P_1 os três métodos têm a mesma performance. para os polinômios P_2, P_3, P_6 e P_7 o método de Halley foi ligeiramente melhor e para os polinômios P_4, P_5 e P_8 o método proposto obteve melhor desempenho.

A menos do polinômio P_1 o método proposto teve melhor desempenho que o método de Newton.



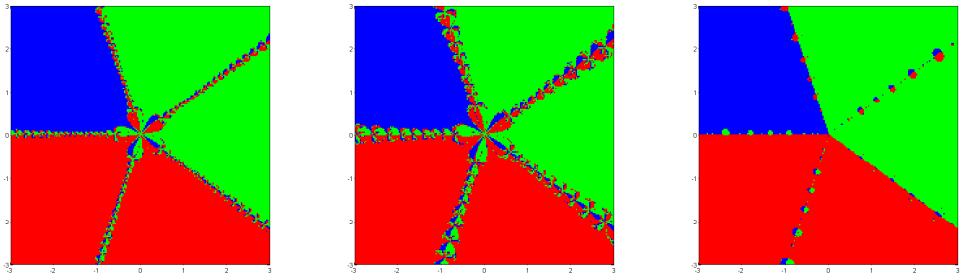
(A) Matioliver (B) Newton (C) Halley

FIGURA 2: Bacias de atração para o polinômio p_1



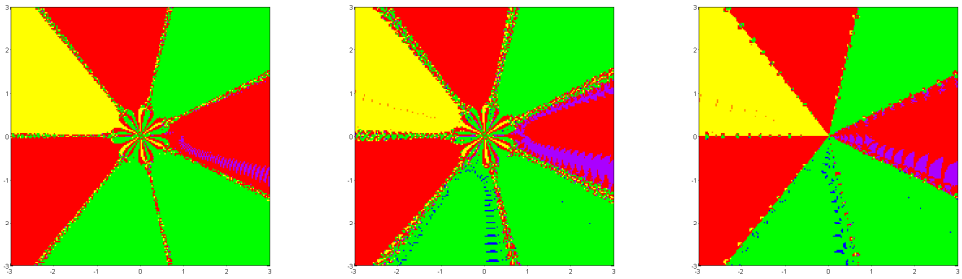
(A) Matioliver (B) Newton (C) Halley

FIGURA 3: Bacias de atração para o polinômio p_2



(A) Matioliver (B) Newton (C) Halley

FIGURA 4: Bacias de atração para o polinômio p_3

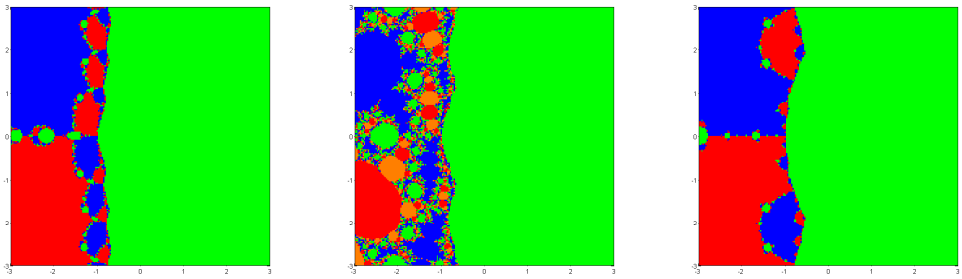


(A) Matioliver (B) Newton (C) Halley

FIGURA 5: Bacias de atração para o polinômio p_4

(A) Matioliver (B) Newton (C) Halley

FIGURA 6: Bacias de atração para o polinômio p_5

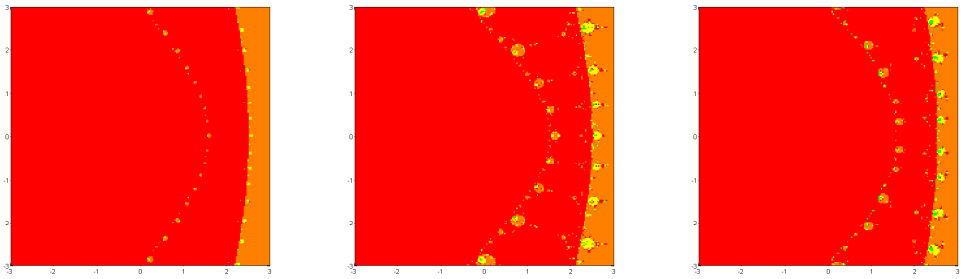


(A) Matioliver (B) Newton (C) Halley

FIGURA 7: Bacias de atração para o polinômio p_6

(A) Matioliver (B) Newton (C) Halley

FIGURA 8: Bacias de atração para o polinômio p_7



(A) Matioliver (B) Newton (C) Halley

FIGURA 9: Bacias de atração para o polinômio p_8

5.2 Conclusão

Neste trabalho desenvolvemos um algoritmo que deflaciona um polinômio, isto é divide-o por um fator $(x-\alpha)^m$, onde m é a multiplicidade conhecida da raiz sem ter que dividir o polinômio pelo fator linear m vezes, e mostramos que este procedimento economiza tempo computacional, no sentido de que ao se aplicar um método para encontrar uma ou duas raízes de um polinômio e combiná-lo com este método, gasta-se menos operações do que se apenas aplicássemos o método para encontrar as raízes e usássemos a deflação usual.

Com base neste método de deflação, bem como utilizando a ideia dos métodos de ponto fixo, introduzimos ainda dois algoritmos para calcular numericamente as raízes de um dado polinômio com coeficientes reais. O primeiro algoritmo determina uma raiz real, enquanto o segundo consegue encontrar tanto raízes reais quanto complexas, e ainda detecta se a raiz encontrada tem multiplicidade 1 ou maior. Outro ponto forte do algoritmo é que se a multiplicidade da raiz encontrada for maior do que

1, os coeficientes que são armazenados no laço interno do algoritmo podem ser usados como os coeficientes do polinômio deflacionado, o que implica em economizar a etapa de deflação. O algoritmo é simples, fácil de implementar e não requer uso de derivadas, tal como nos métodos de Newton e Halley.

Fizemos testes para mostrar a eficácia do algoritmo. As bacias de atração geradas mostram que o algoritmo proposto é promissor e compete com o método de Halley e é ainda melhor que o método de Newton. O método de Newton depende fortemente do ponto inicial para convergir enquanto o método proposto converge independentemente, desde que o ponto inicial seja não nulo. Apesar de convergir para diferentes pontos, conforme sugerem algumas bacias de atração, o método não falha quando mudamos o ponto inicial. O método proposto não faz uso de derivadas, tal como fazem os métodos de Newton e de Halley, com os quais fizemos as comparações.

Sobre considerações futuras, a respeito do método de deflação, nós apenas desenvolvemos métodos que fazem a deflação de um polinômio dado, e através deste desenvolvemos métodos para encontrar uma raiz real ou complexa com certa multiplicidade. Acreditamos que seja possível adaptá-lo para o caso em que houvéssemos encontrado algumas raízes cada uma com um número diferente de multiplicidades, e assim o método efetuaria a deflação de acordo com o fator resultante, digamos $(x-\alpha_1)^{m_1}(x-\alpha_2)^{m_2}...(x-\alpha_k)^{m_k}$. Para isto bastaria generalizar o teorema 3.1.1, e então desenvolver uma adaptação para o caso de raízes complexas.

Já sobre os métodos para cálculo de raízes, poderia-se usar a generalização comentada acima para deduzir um algoritmo que calculasse mais de uma ou duas raízes distintas, baseando-se nas relações encontradas nas generalizações do teorema

3.1.1.

Outros tópicos que podem ser pesquisados futuramente são melhoras nas demonstrações de convergência. Em geral conseguimos mostrar menos do que o método realmente mostra ser capaz de resolver. Por exemplo, não precisamos escalonar o polinômio de modo aos coeficientes satisfazerem as hipótese utilizadas no teorema 4.3.4, o método tem se mostrado o mesmo sem considerá-las. Apesar de as demonstrações da convergência via funções Lipschitzianas requererem que as raízes, bem como os valores iniciais e a sequência gerada pelo algoritmo estejam em uma bola limitada, esta hipótese também não parece ser relevante para o funcionamento do algoritmo. Ainda, as hipóteses (4.6) e (4.8) parecem artificiais, pode ser que seja possível provar que elas sempre valham, ou ainda mostrar outros teoremas sem fazer uso da necessidade destas.

Não chegamos a tentar fazer estas generalizações ou resolver esses problemas, de modo que isto permanece em aberto.

Por fim, gostaríamos de destacar que esta pesquisa está em desenvolvimento. Submetemos um artigo [1] que está em processo de avaliação.

6 APÊNDICE: SEQUÊNCIAS DE NÚMEROS REAIS

Este apêndice é destinado ao leitor que deseja obter uma base de sequências de números reais, cujos resultados são usados com vigor durante o trabalho. As demonstrações aqui encontradas, bem como as definições podem também ser encontradas em um livro de análise como por exemplo, [10].

6.1 Sequências de Números Reais

Nesta seção apresentaremos resultados acerca de sequências de números reais. As demonstrações abaixo são clássicas e podem ser encontradas em um livro de análise, como por exemplo [10].

DEFINIÇÃO 6.1.1 *Uma sequência de números reais é uma função* $f : \mathbb{N} \mapsto \mathbb{R}$.

Usualmente denotamos, para cada $k \in \mathbb{N}$, o valor $f(k)$ por x_k, e a sequência f por $\{x_k\}_{k\in\mathbb{N}}$. Quando claro em contexto, é usual denotar a sequência também por $\{x_k\}$.

DEFINIÇÃO 6.1.2 *Uma subsequência de uma sequência* $\{x_k\}$ *dada por uma função* f *é uma restrição* $f|_{\mathbb{N}'}$ *de* f *a um subconjunto* $\mathbb{N}'$ *infinito de* $\mathbb{N}$.

DEFINIÇÃO 6.1.3 *Dizemos que uma sequência* $\{x_k\}$ *é convergente para* $x \in \mathbb{R}$, *se*

$$\forall \epsilon > 0, \quad \exists n_0 \in \mathbb{N} : k \in \mathbb{N}, k > n_0 \Rightarrow |x_k - x| < \epsilon.$$

DEFINIÇÃO 6.1.4 *Se uma subsequência de* $\{x_k\}$ *converge para um valor* $\overline{x}$*, o que é normalmente denotado por* $x_k \overset{\mathbb{N}'}{\to} \overline{x}$*, então dizemos que* $\overline{x}$ *é um "ponto de aderência", ou "ponto de acumulação" da sequência* $\{x_k\}$*.*

A distinção formal entre os conceitos "ponto de acumulação" e "ponto de aderência" não será aqui apresentada, uma vez que para sequências de pontos distintos as duas noções coincidem. Usaremos aqui ambos os termos indistintamente.

PROPOSIÇÃO 6.1.1 *Se uma sequência* $\{x_k\}$ *é convergente, então possui um único ponto de acumulação.*

Demonstração. Com efeito, se $x_k \to x$, então para todo subconjunto infinito $\mathbb{N}' \subset \mathbb{N}$ temos $x_k \overset{\mathbb{N}'}{\to} x$, donde x é um ponto de aderência de $\{x_k\}$.

Para provar a unicidade, suponha que existam dois valores de aderência x e $\overline{x}$ de $\{x_k\}$. Então existem subconjuntos infinitos $\mathbb{N}'$ e $\mathbb{N}''$ onde

$\forall \epsilon > 0, \quad \exists n_0 \in \mathbb{N}', \quad \exists n_1 \in \mathbb{N}''$ tal que $k > n_0, n_1 \Rightarrow |x_k - x| < \epsilon/2, \quad |x_k - \overline{x}| < \epsilon/2.$

Sendo assim, pela desigualdade triangular para módulos, $|x - \overline{x}| \leq |x - x_k| + |x_k - \overline{x}| \leq \epsilon/2 + \epsilon/2 = \epsilon$. Como isto vale para todo ϵ, segue que $x = \overline{x}$. ∎

DEFINIÇÃO 6.1.5 *Uma sequência é dita limitada, se existe* $M > 0$ *tal que para todo* $k \in \mathbb{N}$*, tem-se* $|x_k| \leq M$*.*

DEFINIÇÃO 6.1.6 *Uma sequência* $\{x_k\}$ *é dita monótona crescente se para todo* $k \in \mathbb{N}$ *vale* $x_k < x_{k+1}$*.*

Analogamente, dizemos que a sequência $\{x_k\}$ é monótona decrescente, não

decrescente, ou não crescente, trocando-se o sinal $<$ na definição por respectivamente $>$, $\leq$ e $\geq$. Em qualquer um dos quatro casos $\{x_k\}$ é dita “monótona”.

PROPOSIÇÃO 6.1.2 *Toda sequência convergente é limitada*

Demonstração. De fato, seja $\{x_k\} \to x$, então

$$\forall \epsilon > 0, \quad \exists n_0 \in \mathbb{N} \text{ tal que } k > n_0 \Rightarrow |x_k| - |x| < |x_k - x| < \epsilon.$$

assim, $|x_k| < \epsilon + |x|$. Os $n_0 - 1$ primeiros elementos de $\{x_k\}$ são limitados por eles próprios, assim, toda a sequência $\{x_k\}$ fica limitada por

$0 < M = max\{|x_1|, \ldots |x_{n_0-1}|, \epsilon + |x|\}$. ∎

PROPOSIÇÃO 6.1.3 *Toda sequência monótona e limitada é convergente.*

Demonstração. À luz da proposição (6.1.1), fixaremos uma sequência $\{x_k\}$ e mostraremos que ela possui um único ponto de acumulação. Para provar a unicidade, suponha que $\{x_k\}$ possua mais de um ponto de aderência, digamos, $\{x_k\} \overset{\mathbb{N}'}{\to} x1$, e $\{x_k\} \overset{\mathbb{N}''}{\to} x2$, então supondo por exemplo, $\{x_k\}$ crescente, e $x1 < x2$, tomando $\epsilon = \frac{x2+x1}{2}$, temos que existe $n_0 \in \mathbb{N}$ tal que se $k > n_0$, vale $|x1 - x_k| < \epsilon$, para todo $k \in \mathbb{N}'$, e $|x2 - x_k| < \epsilon$, para todo $k \in \mathbb{N}''$, donde $x_k < \frac{x2+x1}{2}$, se $k \in \mathbb{N}'$, e $x_k > \frac{x1+x2}{2}$, quando $k \in \mathbb{N}''$, o que contradiz a monotonicidade de $\{x_k\}$.

para provar a existência do ponto de aderência, suponha sem perda de generalidade que $\{x_k\}$ seja crescente, então o ínfimo de suas cotas superiores é um ponto de aderência, com efeito, seja M tal número. Pela definição de ínfimo, temos que para todo $\epsilon > 0 \quad \exists \quad x_{n_\epsilon}$ tal que $|M - x_{n_\epsilon}| < \epsilon$. Em particular tomando-se $\epsilon = \frac{1}{k}$, conseguimos uma subsequência $\{x_{n_k}\}$ tal que $|M - x_{n_k}| < \frac{1}{k}$, de modo que M é um ponto de aderência, o que prova a existência. ∎

Outro resultado clássico, que será utilizado posteriormente é o teorema de Bolzano-Weirstrass:

TEOREMA 6.1.4 (BOLZANO-WEIRSTRASS) *Toda sequência limitada de números reais admite subsequência convergente*

Demonstração. Seja $\{x_k\}$ a sequência em questão. Primeiro, defina $x_{n_1} = x_1$. Como a sequência é limitada, existe um intervalo $[a_1, b_1] \subset \mathbb{R}$ tal que a sequência toda está contida neste. Seja M_1 o ponto médio entre a_1 e b_1. Então tem-se que $[a_1, b_1] = [a_1, M_1] \cup [M_1, b_1]$, e portanto deve haver pelo menos um destes intervalos com a propriedade que infinitos elementos de $\{x_k\}$ pertencem à ele. Escolha um destes intervalos. Defina x_{n_2} como qualquer elemento da sequência que pertence ao intervalo escolhido contanto que $n_2 > n_1$. Se o intervalo escolhido foi aquele que fica à direita, então defina: $a_2 = M_1$, e $b_2 = b_1$. Caso contrário escolha: $a_2 = a_1$, e $b_2 = M_1$. Observe que por construção, o comprimento do intervalo foi reduzido pela metade. Repita este processo recursivamente, de forma a obter uma sequência de intervalos $[a_k, b_k]$ e de pontos da sequência $\{x_{n_k}\}$ com as seguintes propriedades: $x_{n_k} \in [a_k, b_k]$, $b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2}$, $a_k \geq a_{k-1}$, e $b_k \leq b_{k-1}$.

Assim, a_k é uma sequência de números reais não decrescente limitada superiormente por b_1, e logo, pela proposição (6.1.3), converge para um limite a.

Analogamente b_k é uma sequência não-crescente e limitada inferiormente por a_1, portanto também converge para um limite b. Como $b_k - a_k \to 0$, pois o comprimento dos intervalos é reduzido pela metade a cada vez, tem-se que $a = b$. Como $x_{n_k} \in [a_k, b_k]$,segue que x_{n_k} converge para $a = b$, e logo é uma subsequência convergente.

■

DEFINIÇÃO 6.1.7 *Uma sequência* $\{x_k\}$ *é dita "de Cauchy" se*

$$\forall \epsilon > 0, \quad \exists n_0 \in \mathbb{N} \textit{ tal que } k1, k2 > n_0 \Rightarrow |x_{k1} - x_{k2}| < \epsilon.$$

Para sequências de números reais as afirmações "ser convergente" e "ser de Cauchy" são equivalentes, com efeito, se $\{x_k\}$ for convergente, então para todo $\epsilon > 0$, existe $n_0 \in \mathbb{N}$ tal que $k1 > n_0 \Rightarrow |x_{k1} - x| < \frac{\epsilon}{2}$, e $k2 > n_0 \Rightarrow |x_{k2} - x| < \frac{\epsilon}{2}$, e portanto, pela desigualdade triangular para módulos, tem-se $|x_{k1} - x_{k2}| < |x_{k1} - x| + |x_{k2} - x| < 2\frac{\epsilon}{2} = \epsilon$.

Por outro lado, se $\{x_k\}$ for "de Cauchy", então é limitada, pois fixado $\epsilon > 0$, existe $n_0 \in \mathbb{N}$ tal que

$$k1, k2 > n_0 \Rightarrow |x_{k1} - x_{k2}| < \epsilon.$$

Em particular, $|x_{k_{n_0+1}} - x_{k1}| < \epsilon$, donde $|x_{k1}| < \epsilon + |x_{n_0+1}|$, para todo $k1 > n_0$. Tomando $M = max\{x_1, \ldots x_{n0}, |x_{n_0+1}| + \epsilon\}$, temos a constante que limita $\{x_k\}$.

Além disso, a sequência $\{x_k\}$ não pode ter dois pontos de aderência $x1 < x2$ distintos, pois se assim fosse, fazendo $\epsilon = \frac{x1+x2}{2}$, existiria $n_0 \in \mathbb{N}$ tal que se $k > n_0$, então

$$|x_k - x1| < \epsilon, \text{ se } k \in \mathbb{N}', \text{ e } |x_l - x2| < \epsilon, \text{ se } l \in \mathbb{N}''.$$

donde para $k > n_0 \in \mathbb{N}'$, e $l > n_0 \in \mathbb{N}''$, teríamos

$$|x_k - x_l| \geq |x_l| - |x_k| \geq \epsilon,$$

o que contradiz a definição de sequência de Cauchy. Segue que $\{x_k\}$ é limitada e tem no máximo um ponto de acumulação, donde deve ser convergente.

Todos os resultados aqui enunciados para sequências de números reais podem ser estendidos para sequências de vetores de $\mathbb{R}^n$. Para sequências em $\mathbb{R}^n$, denotaremos por $x_k \in B(x, \epsilon)$ a propriedade $\|x_k - x\| < \epsilon$, onde $\|.\|$ é a norma euclidiana de $\mathbb{R}^n$.

Também precisaremos estender o teorema de Bolzano-Weirstrass para números complexos. Para tal precisaremos da seguinte definição:

DEFINIÇÃO 6.1.8 *Seja $\{x_k + iy_k\}$ uma sequência de números complexos, isto é, x_k e y_k são reais, e i é a unidade imaginária para todo elemento $x_k + iy_k$ em $\{x_k + iy_k\}$. Dizemos que $\{x_k + iy_k\}$ é limitada se existem intervalos $[a, b]$ e $[c, d]$ em $\mathbb{R}$ tais que $x_k \in [a, b]$, $y_k \in [c, d]$ $\forall$ $x_k + iy_k \in \{x_k + iy_k\}$. Sem perda de generalidade podemos dizer o mesmo se existe um quadrado em $\mathbb{R} \times \mathbb{R}$ tal que a sequência $\{x_k + iy_k\}$ está contida neste quadrado.*

TEOREMA 6.1.5 (BOLZANO-WEIRSTRASS PARA NÚMEROS COMPLEXOS) *Toda sequência limitada de números complexos admite subsequência convergente*

Demonstração. Seja $\{x_k + iy_k\}$ uma sequência limitada de números complexos. Então da limitação, existe um quadrado em $\mathbb{R}\times\mathbb{R}$ donde a sequência está contida neste quadrado. Este quadrado pode sem perda de generalidade ser tomado como o cartesiano $[a_1, b_1] \times [a_1, b_1]$, de modo que $x_k \in [a_1, b_1]$ e $y_k \in [a_1, b_1]$ com $[a_1, b_1] \subset \mathbb{R}$. Aplicando o mesmo raciocínio usado no teorema (6.1.4) para x_k vemos que esta possui subsequência convergente, digamos $x_k \overset{\mathbb{N}'}{\to} x$, e como $\mathbb{N}'$ é um subconjunto infinito de $\mathbb{N}$, podemos usar o mesmo raciocínio para a sequência restrita $\{y_k\}_{k\in\mathbb{N}'}$, de modo a encontrar um subconjunto infinito $\mathbb{N}'' \subset \mathbb{N}'$ tal que $y_k \overset{\mathbb{N}''}{\to} y$, e pela ordem de inclusão, da mesma forma tem-se $x_k \overset{\mathbb{N}''}{\to} x$. Sendo assim, temos que $x_k + iy_k \overset{\mathbb{N}''}{\to} x + iy \in \mathbb{C}$, e

portanto $x_k + iy_k$ possui subsequência convergente. ∎

Referências

[1] O. Kolossoski, L. C. Matioli, *A Deflation Based Method for Finding Roots of a Polynomial*, Submetido em 2012.

[2] B. Neta, *Extension of Murakami's high-order non-linear solver to multiple roots*, International Journal of Computer Mathematics, 87-5(2010), 1023-1031.

[3] B. Neta, *New third order nonlinear solvers for multiple roots*, Appl. Math. Comput. 202 (2008), 162-170.

[4] M. Scott, B. Neta, C. Chun, *Basin attractors for various methods*, Appl. Math. Comput., 218 (2011) 2584-2599.

[5] B. Neta, M. Scott, C. Chun, *Basin attractors for various methods for multiple roots*, Appl. Math. Comput. 218 (2012) 5043-5066.

[6] E. Halley, *A new, exact and easy method for finding the roots of equations generally and that without any previous reduction*, Philos. Trans. R. Soc. Lond. 18(1694) 136-148.

[7] B.D. Stewart, *Attractor Basins of Various Root-Finding Methods*, M.S. thesis, Naval Postgraduate School, Department of Applied Mathematics, Monterey, CA, June 2001.

[8] G.W. Stewart, *The Convergence of Multipoint Iterations to Multiple Zeros*, SIAM Journal on Numerical Analysis, 11(1974) 1105-1120.

[9] P. Henrici, *Elements of Numerical Analysis*, John Wiley & Sons Inc, 1964.

[10] E. L. Lima, *Curso de Análise, Vol. 1, 12Ed.*, SBM/IMPA, 2006.

[11] M. Minoux, *Mathematical Programming*, Wiley-Interscience, 1986.

[12] D. Goldberg, *What Every Computer Scientist Should Know About Floating Point Arithmetic*, ACM surveys 23, 5-48.

[13] M.A. Jenkins, and J.F. Traub, *A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration*, SIAM Journal on Numerical Analysis, 7(1970) 545-566.

[14] M.A. Jenkins, and J.F. Traub, *Principles for Testing Polynomial Zerofinding Programs*, ACM Trans. Math. Sof., 1(1975) 26-34.

[15] John Michael McNamee, *An updated supplementary bibliography on roots of polynomials*, Journal of Computational and Applied Mathematics, 110(1999) 305-306.

[16] Victor Y. Pan, *Some History and Recent Progress*, SIAM Review, 39-2(1997) 187-220.

[17] Zhonggang Zeng, *Computing Multiple Roots of Inexact Polynomials*, Mathematics of Computation, 74(2004) 869-903.

[18] J.A. Stolan, *An improves $\check{S}$iljak's algorithm for solving polynomials equations converges quadratically to multiple zeros*, Journal of Computational and Applied Mathematics, 64(1995) 247-268.

[19] Chang-Dau Yan, and Wei-Hua Chieng, *Method for Finding Multiple Roots of Polynomials*, Computers and mathematics with Applications, 51(2006) 605-620.

[20] V. Hribernig, and H.J. Stetter, *Detection and Validation of Clusters of Polynomial Zeros*, J. Symbolic Computation, 24(1997) 667-682.

[21] F.M. Carrano, *A Modified Bairstow Method for multiple Zeros of a Polynomial*, Mathematics of Computation, 27(1973) 781-792.